

# Long-Horizon Direct Model Predictive Control of an Active Capacitor for Ripple Energy Compensation in Single-Phase DC-to-AC Converters

by

Macyln Tatenda Chingwena



*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 2021

# 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.

Macyln Tatenda Chingwena

Copyright © 2021 Stellenbosch University  
All rights reserved

# Abstract

Single-phase dc-to-ac converters generate power on the ac side that pulsates at twice the grid frequency. Inherently, the pulsating power is transferred from the ac side to the dc side and generates second-order harmonic currents that flow through the dc bus, also referred to as the ripple current. This occurs when power flows either from the ac side to the dc side or when power flows from the dc side to the ac side. In this thesis, we assume power flow from the dc side to the ac side.

Suppose a battery powers a single-phase dc-to-ac converter that is connected to either the grid or a load. The generated ripple current will unavoidably flow through the battery. Although ideally the current flowing through a battery should be constant, that is nearly impossible. Generally, the ripple current passing through the battery should be limited to 10 % of the nominal battery current.

Usually, a dc-link capacitor is used to reduce the ripple current, and aluminium electrolytic capacitors are often used due to their availability in large capacitance values. However, they have a short lifespan and bulk size, which leads to reliability issues. This creates a trade-off between reducing either the ripple current or the capacitance requirements.

The concept of using an energy storage circuit for ripple energy compensation and, at the same time, reducing the capacitance requirements has been proposed. However, a control problem is formulated when using this method of ripple energy compensation. The ripple current needs to be diverted away from the battery to the energy storage circuit. In previous years, classical control strategies were used to address the control problem. Nonetheless, the closed-loop performance of these controllers still presents challenges.

The main contribution of this thesis is on using model predictive control to compensate for ripple energy, with a dc-to-dc boost converter as an energy storage circuit. Since model predictive control has only recently been adopted in power electronics, it still bears a stigma that longer prediction horizons do not offer performance benefits. In this thesis, the implementation of long-horizon direct model predictive control for a boost converter is given in great detail. By using the branch-and-bound technique and the move blocking strategy, the optimization problem is solved efficiently, enabling practical considerations.

Through simulations, the efficacy of the control strategy is verified. For horizons less than three, the system did not reach steady-state operation, validating the need for longer prediction horizons. It is shown that, for a prediction horizon of ten, the ripple current is reduced to 2.4 % and 2.8 % of the nominal battery current, for a grid-connected system and a stand-alone system, respectively. At the same time, the capacitance requirements are reduced by over 95 % for both systems.

The controller is implemented within a field-programmable gate array, and through a hardware-in-the-loop simulation of a stand-alone system, the practical feasibility of the controller is verified. It is shown that the ripple current is reduced to roughly 3.2 % of the nominal battery current when using a prediction horizon of seven.

# Opsomming

Enkelfase DS-na-WS omsetters genereer drywing aan die WS kant wat puls teen twee maal die kragnetwerk se frekwensie. Die pulserende drywing word inherent oorgedra vanaf die WS kant van die omsetter na die DS kant. Hierdie proses genereer tweede-orde harmonieke strome, bekend as die rimpelstroom, wat vloei deur die DS bus. Dit vind plaas wanneer die drywing vloei vanaf die WS-na-DS kant of andersom.

Gestel dat 'n battery 'n enkelfase DS-na-WS omsetter aandryf wat gekoppel is aan óf die kragnetwerk óf 'n las. Die rimpelstroom wat gegenereer word sal onvermydelik deur die battery vloei. Alhoewel die stroom wat deur 'n battery vloei konstant behoort te wees, is dit byna onmoontlik in praktyk. Die rimpelstroom wat deur die battery vloei moet tipies beperk word tot 10 % van die nominale batterystroom.

Normaalweg sal 'n DS-skakel kapasitor gebruik word om die rimpelstroom te beperk. Aluminium elektrolitiese kapasitore word dikwels vir hierdie toepassing gebruik aangesien dit beskikbaar is in hoë kapasitansie waardes. Hierdie kapasitore het egter 'n kort lewensduur en 'n ongemaklike grootte wat lei tot onbetroubaarheid. Die gevolg is 'n kompromie tussen die vermindering van die rimpelstroom en die kapasitansievereistes.

Die konsep van 'n energie-bergende stroombaan word voorgestel wat rimpel-energie kan onderdruk sowel as die kapasitansie vereisters verminder. Daar word egter 'n beheer-probleem geskep met so 'n voorstel van rimpel-energie onderdrukking. Die rimpelstroom moet weggelei word vanaf die battery na die energie-bergende stroombaan. Voorheen was klassieke beheertegniese gebruik om hierdie beheer-probleem aan te spreek. Die geslotelus optrede van hierdie beheerstelsels bied egter steeds uitdagings.

Die hoof bydrae van hierdie tesis is om voorspellende beheer te gebruik om te kompenseer vir die rimpel-energie, terwyl 'n DS-na-DS opkapper gebruik word as die energie-bergende stroombaan. Aangesien voorspellende beheer eers onlangs aangepas is vir elektronika, dra dit steeds die stigma dat langer horisonne nie 'n beduidende voordeel bied vir die uittree-optrede nie. In hierdie tesis word die implentering van 'n lang-horison voorspellende beheerder in detail weergegee. Deur gebruik te maak van die tak-en-gebonde tegniek sowel as die beweg-bokkeerstrategie, kan die optimeringsprobleem doeltreffend opgelos word. Dit lei tot die oorweging van praktiese implenterings.

Die doeltreffendheid van die beheerstrategie word bevestig deur simulاسies. Vir horisonne korter as drie het die stelsel nie bestendigetoestand werking bereik nie, wat die vereiste van langer horisonne aandui. Dit word gewys dat die rimpelstroom verminder word vir die nominale batterystroom vir 'n horison van tien tot 2.4 % vir 'n kragnetwerk-gekoppelde stelsel en tot 2.8 % vir 'n alleenstaande stelsel. Op dieselfde tyd word die kapasitansie vereistes verminder tot 95 % vir beide stelsels.

Die beheerder word geïmplementeer op 'n veld-programmeerbare hekskikking (FPGA). Die praktiese uitvoerbaarheid van die beheerder word bevestig deur middel van 'n hardware-in-die-lus simulاسie van die alleenstaande stelsel. Dit word gewys dat die rimpelstroom verminder word tot 3.2 % van die nominale batterystroom met 'n voorspellingshorison van sewe.

# Acknowledgments

In the words of Peter S. Beagle, Tamsin “*But what I thought, and what I still think, and always will, is that she saw me. Nobody else has ever seen me, me!, ... Love is one thing, yes, but recognition is something else.*” So here’s to all that recognized me throughout my Master’s journey specifically mentioned or not.

To Prof Mouton, you have been my supervisor since my final undergraduate year, and you never ceased to encourage and believe in me in my pursuit of my Master’s degree. Be known to you, I stumbled along the way, but you always showered me with words of affirmation and encouragement when the load became a little too overwhelming. You were attentively available to help me with anything every single step of the way; and your vast knowledge and intellect in Power Electronics was and still is beyond impressively imaginable. From the battles you fought behind the scenes for my funding, to your great sense of humor and the treats you used to prepare for us (to be more specific, let’s not forget that apple crumble pie “*that only your students ate*”) I will forever be grateful to you. Baie dankie, Prof.

To my mom, Sylvia, you were my support system throughout the journey. Without your hard work that saw me, and supported me throughout my undergraduate studies, I wouldn’t have managed to get the opportunity to do my Master’s. I love you, Mom.

To my young brother, Lesley, your selflessness is admirable beyond measure. You most importantly encouraged me to push the pause button and to just relax for a moment. You forced me to watch episodes together of some of the best series ever created, such as the amazing Game of Thrones (season finale, although it was such a disappointment by the way), Ozark, Breaking Bad, and Peaky Blinders, all because you understood that it was okay for me to stop, relax then continue with my work because mental health was equally important in my Master’s pursuit. With the late nights at the engineering faculty, I always returned to a fresh homemade meal, and sometimes a packed lunch for the next day. I am extremely grateful Lele, more than you will ever know. You kept me sane, centered and focused during the lock-down of this global pandemic and above all made it bearable.

To Lauren, I really appreciate how you meticulously proofread my thesis and gave me valuable feedback. Besides that, you were very supportive with regards to my mental well-being. Thank you my friend.

I extend my gratitude to all my friends, family members, and university staff who I did not specifically mention by name. I sincerely appreciate all of your support. It has been an incredible journey of highs and lows and, in the end, it was worth it, because it counted. Cheers and thank you.

# Special acknowledgment

Tinus Dorfling

Honestly, words failed me in trying to describe your contribution to my Master's, Tinus. You became more of a mentor, and a friend; you were always willing to lend an ear and offer advice to solve any challenge I faced. *To you, I dedicate this thesis.*

# Contents

Declaration	i
Abstract	ii
Opsomming	iii
Special acknowledgment	v
List of figures	ix
List of tables	xi
Nomenclature	xii
<b>1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	1
1.2 Thesis objectives . . . . .	5
1.3 Thesis outline . . . . .	6
<b>2 Theoretical background</b>	<b>8</b>
2.1 The single-phase dc-to-ac converter . . . . .	8
2.1.1 Power semiconductor devices . . . . .	8
2.1.2 Introduction to topology . . . . .	9
2.1.3 Carrier-based pulse-width modulation . . . . .	10
2.1.4 Ripple power in a single-phase dc-to-ac converter . . . . .	18
2.1.5 Analysis of capacitive ripple energy compensation . . . . .	20
2.2 Active capacitor for ripple energy compensation . . . . .	25
2.2.1 Analysis of dc-to-dc converter topologies . . . . .	25
2.2.2 The operation principle of the dc-to-dc boost converter and control problem formulation . . . . .	26
2.2.3 Proposed control schemes . . . . .	28
2.3 Model predictive control . . . . .	29
2.3.1 Introduction to MPC . . . . .	29
2.3.2 Internal dynamic model . . . . .	30
2.3.3 Constraints . . . . .	31
2.3.4 Optimal control problem . . . . .	32
2.3.5 Receding horizon policy . . . . .	34
2.3.6 MPC based on exhaustive search . . . . .	35
2.3.7 Computational burden . . . . .	35

2.4	Branch-and-bound technique . . . . .	36
2.4.1	Introduction to the branch-and-bound technique . . . . .	36
2.4.2	The initial upper bound guess . . . . .	36
2.4.3	Illustrative example of the branch-and-bound technique . . . . .	37
2.5	Move blocking strategy . . . . .	39
2.5.1	Introduction to the move blocking strategy . . . . .	39
2.5.2	Implementation of the move blocking strategy . . . . .	39
2.5.3	Illustrative example of the move blocking strategy . . . . .	39
<b>3</b>	<b>Model predictive control of a boost converter</b>	<b>41</b>
3.1	Introduction . . . . .	41
3.2	Modelling and control law formulation . . . . .	41
3.3	Optimal control of the boost converter . . . . .	44
3.3.1	Control of the active capacitor voltage . . . . .	44
3.3.2	Control algorithm . . . . .	47
3.3.3	Generation of the reference ripple current . . . . .	51
3.4	Summary . . . . .	52
<b>4</b>	<b>Performance evaluation of long prediction horizons</b>	<b>53</b>
4.1	Introduction . . . . .	53
4.2	Procedure for performing simulations . . . . .	53
4.3	Steady-state operation . . . . .	54
4.3.1	Grid-connected system . . . . .	54
4.3.2	Stand-alone system . . . . .	60
4.4	Response during transients . . . . .	64
4.5	Summary . . . . .	69
<b>5</b>	<b>FPGA implementation</b>	<b>71</b>
5.1	Introduction . . . . .	71
5.2	The FPGA . . . . .	71
5.2.1	A brief description of the FPGA board . . . . .	71
5.2.2	FPGA building blocks . . . . .	71
5.2.3	FPGA preliminaries . . . . .	73
5.3	VHDL implementation . . . . .	74
5.3.1	Arithmetic in VHDL . . . . .	74
5.3.2	Computation delay compensation . . . . .	75
5.3.3	Hardware-in-the-loop simulation . . . . .	78
5.3.4	Controller implementation . . . . .	82
5.3.5	Computational burden . . . . .	85
5.3.6	VHDL verification . . . . .	87
5.4	Summary . . . . .	91
<b>6</b>	<b>Conclusions</b>	<b>92</b>
6.1	Overview of results . . . . .	92
6.1.1	Performance evaluation of long prediction horizons . . . . .	92
6.1.2	FPGA implementation . . . . .	93
6.2	Recommendations for future research . . . . .	94
6.2.1	Tuning of parameters . . . . .	94
6.2.2	Active capacitor voltage reference . . . . .	95



<i>CONTENTS</i>	viii
6.2.3 Experimental tests . . . . .	95
<b>Appendices</b>	<b>96</b>
<b>A Mathematical preliminaries</b>	<b>97</b>
<b>B Ripple power analysis of a single-phase dc-to-ac converter with an <math>RL</math> load</b>	<b>98</b>

# List of figures

1.1	A comparison of different battery technologies. . . . .	2
1.2	Block diagram illustrating power transfer in a single-phase dc-to-ac converter. . . . .	2
1.3	Block diagram showing the conventional method of ripple energy compensation. . . . .	3
1.4	Trade-off between the ripple current and total capacitance required. . . . .	3
1.5	Block diagram showing the concept of using an energy storage circuit. . . . .	4
1.6	Ripple current rating and the required capacitance. . . . .	4
2.1	The basic structure of a full-bridge single-phase dc-to-ac converter. . . . .	9
2.2	Example of pulse-width modulation with bipolar switching. . . . .	11
2.3	Example of pulse-width modulation with unipolar switching. . . . .	13
2.4	Positive current paths of the single-phase dc-to-ac converter switches. . . . .	15
2.5	Negative current paths of the single-phase dc-to-ac converter switches. . . . .	15
2.6	Harmonic spectrum of the switch positions $u_{sw,A}(t)$ . . . . .	16
2.7	Harmonic spectrum of the switching function $u_{pwm}(t)$ . . . . .	18
2.8	Grid-connected single-phase dc-to-ac converter with a dc-link capacitor. . . . .	18
2.9	Example that illustrates ripple power. . . . .	19
2.10	The key waveforms for ripple energy analysis. . . . .	21
2.11	The concept of using an energy storage circuit for ripple energy compensation. . . . .	23
2.12	The circuit configurations of dc-to-dc converters. . . . .	24
2.13	Proposed method of ripple energy compensation. . . . .	26
2.14	Charging mode of the active capacitor. . . . .	27
2.15	Discharging mode of active capacitor. . . . .	27
2.16	Block diagram of the basic control loop. . . . .	31
2.17	Example of the receding horizon policy for a prediction horizon of $N_p$ . . . . .	34
2.18	Example of a search tree with depth $n = N_p$ . . . . .	36
2.19	Example of the transversal of a search tree with depth $n = 4$ . . . . .	37
2.20	Example of prediction horizon without move blocking. . . . .	40
2.21	Example of prediction horizon with move blocking. . . . .	40
3.1	Topology of the dc-to-dc boost converter. . . . .	41
3.2	Automation of the boost converter continuous-time state-space representation. . . . .	42
3.3	Automation of the boost converter discrete-time state-space representation. . . . .	43
3.4	Block diagrams of control loops for the boost converter. . . . .	44
3.5	Illustrative example of the effectiveness of the move blocking strategy. . . . .	46
3.6	Backtracking example. . . . .	50
3.7	Block diagram of the proposed MPC strategy. . . . .	51
3.8	Frequency response of the band-pass filter. . . . .	51
3.9	Illustration of the generation of the reference ripple current. . . . .	52

4.1	Simulation waveform of the rectified current for a grid-connected system. . . . .	55
4.2	Simulation waveforms of the inductor current and active capacitor voltage of the boost converter with $N_p = 10$ . . . . .	56
4.3	Simulation waveform of the battery current for the grid-connected system with $N_p = 10$ . . . . .	57
4.4	Harmonic amplitude spectrum of the battery current for the grid-connected system with $N_p = 10$ . . . . .	58
4.5	Simulation waveform of the rectified current for a stand-alone system. . . . .	60
4.6	Simulation waveforms of the inductor current and capacitor voltage of the boost converter with $N_p = 10$ . . . . .	61
4.7	Simulation waveform of the battery current for the stand-alone system with $N_p = 10$ . . . . .	62
4.8	Harmonic amplitude spectrum of the battery current for the stand-alone system with $N_p = 10$ . . . . .	63
4.9	Simulation waveform of the load current $i_g(t)$ with a step-up change. . . . .	65
4.10	The transient response of the inductor current $i_L(t)$ for a step-up change in the load current. . . . .	66
4.11	Simulation waveform of the active capacitor voltage $v_c(t)$ for a step-up change in the load current. . . . .	66
4.12	Simulation waveform of the battery current for a step-up change in the load current. . . . .	67
4.13	Simulation waveform of the load current $i_g(t)$ with a step-down change. . . . .	67
4.14	The transient response of the inductor current $i_L(t)$ for a step-down change in the load current. . . . .	68
4.15	Simulation waveform of the active capacitor voltage $v_c(t)$ for a step-down change in the load current. . . . .	68
4.16	Simulation waveform of the battery current for a step-down change in the load current. . . . .	69
5.1	Example of the ideal scenario with zero computation delay. . . . .	75
5.2	Example of the practical scenario with a computation delay time. . . . .	76
5.3	Illustration of delay compensation. . . . .	77
5.4	Topology of the stand-alone system. . . . .	78
5.5	Output stage of the single-phase dc-ac converter. . . . .	79
5.6	Input stage of the single-phase dc-ac converter. . . . .	80
5.7	An automation of the triangular signal generation procedure. . . . .	82
5.8	Triangular signal generation on an FPGA. . . . .	83
5.9	Placing of signals in an array according to the move blocking strategy. . . . .	84
5.10	A finite-state machine (FSM) that monitors the number of clock cycles. . . . .	86
5.11	Approximated number of clock cycles required to obtain the optimal solution. . . . .	87
5.12	FPGA and MATLAB® simulations comparison of the inductor current. . . . .	88
5.13	FPGA and MATLAB® simulations comparison of the active capacitor voltage. . . . .	89
5.14	Harmonic amplitude spectrum of the current through the battery $i_b(t)$ before ripple energy compensation on the FPGA. . . . .	89
5.15	Harmonic amplitude spectrum of the current through the battery $i_b(t)$ after ripple energy compensation on the FPGA. . . . .	90
B.1	Topology of the stand-alone system. . . . .	98

# List of tables

2.1	Switching states for pulse-width modulation with bipolar switching. . . . .	12
2.2	Switching states for pulse-width modulation with unipolar switching. . . . .	16
2.3	Parameters of a single-phase dc-to-ac converter. . . . .	22
4.1	Grid-connected system simulation parameters. . . . .	54
4.2	Comparison of the second-order harmonic component at $f_{sw} \approx 16$ kHz. . . . .	59
4.3	Stand-alone system simulation parameters. . . . .	60
4.4	Comparison of the second-order harmonic component at $f_{sw} \approx 16$ kHz . . . . .	64
5.1	FPGA and MATLAB® simulations comparison of the second-order harmonic component at $f_{sw} \approx 16$ kHz. . . . .	91

# Nomenclature

## Acronyms

ac	alternating current
ADC	analogue-to-digital converter
AE	aluminium electrolytic
AP	all programmable
BJT	bipolar junction transistor
CB	carrier-based
CCM	continuous conduction mode
CCS	continuous-control set
CLB	configurable logic block
dc	direct current
DSP	digital signal processor
EMI	electromagnetic interference
FCS	finite-control set
FFT	fast Fourier transformation
FPGA	field-programmable gate array
FSM	finite-state machine
HDL	hardware description language
HiL	hardware-in-the-loop
IDE	integrated development environment
IGBT	insulated-gate bipolar transistor
ILA	integrated logic analyzer
I/O	input/output
LED	light emitting diode
LFP	lithium-ion phosphate
LIFO	last-in, first-out
Li-ion	lithium-ion
LUT	lookup table

## Acronyms

MIMO	multiple-input multiple-output
MOSFET	metal-oxide-semiconductor field-effect transistor
MPC	model predictive control
MPPF	metallized polypropylene film
Ni-Cd	nickel-cadmium
Ni-MH	nickel-metal hydride
PI	proportional-integral
PV	photovoltaic
PWM	pulse-width modulation
RAM	random-access memory
ROM	read-only memory
SISO	single-input single-output
SoC	system on a chip
THD	total harmonic distortion
VHDL	VHSIC hardware description language
VHSIC	very high speed integrated circuit

## Symbols

$\lambda_1, \lambda_2$	weighting factors on tracking error
$\lambda_u$	weighting factor on switching effort
$f_1$	fundamental frequency [Hz]
$f_r$	frequency of second-order harmonic component [Hz]
$f_c$	switching frequency [Hz], PWM
$f_{sw}$	switching frequency [Hz], MPC
$f_{hil}$	sampling frequency [Hz], HiL
$f_{clk}$	clock frequency [Hz], FPGA
$i, I$	ampere [A]
$\mathbf{I}_n$	identity matrix with dimension $n$
$J$	objective function of the optimization problem
$k$	discrete time step, $k \in \mathbb{N}$
$l$	discrete time steps in a prediction horizon, $l \in \{k, k + 1, \dots, k + N_p\}$
$L$	inductance [H]
$m_a$	modulation index
$N_p$	prediction horizon
$n_u, n_x, n_y$	size of input-, state-, and output variables
$\mathbb{N}^+$	positive integers
$\emptyset$	empty set
$Q$	penalty matrix

## Symbols

$R$	resistance [ $\Omega$ ]
$t$	Time, $t \in \mathbb{R}^+$
$T_c$	sampling interval [s], PWM
$T_s$	sampling interval [s], MPC
$T_{hil}$	sampling interval [s], HiL
$T_{ref}$	sampling interval [s], PWM reference signals for HiL
$\omega_1$	angular fundamental frequency, $\omega_1 = 2\pi f_1$
$u, \mathbf{u}$	switch position (input variable)
$\Delta u, \Delta \mathbf{u}$	switching transition
$\mathbf{U}$	switching sequence over prediction horizon
$\mathcal{U}$	$\{0, 1\}$ , set of admissible switch positions
$v, V$	voltage [V]
$x$	state variable
$y$	output variable

## Operators

$\dot{x}$	$\frac{dx}{dt}$ , derivative of $x$ with respect to time
$x \in \mathcal{X}$	$x$ belongs to set $\mathcal{X}$

# Chapter 1

## Introduction

### 1.1 Background

In the past decades, electricity was generated mostly from conventional energy sources such as fossil fuels. Owing to the environmental and geopolitical concerns of conventional energy sources, their use has continued to decline in recent years. Renewable energy resources, on the other hand, are becoming the preferred option for electricity generation for both grid-connected and stand-alone systems since they minimize the threat of global warming, climate change, and pollution to the environment. In particular, wind and solar power have become the most significant renewable energy resources with a high annual growth rate in wind turbine and photovoltaic (PV) array installations.

Despite the advantages of renewable energy sources, they cannot continuously supply a base-load due to their intermittent nature. Therefore, to address this issue, energy storage systems can be introduced to partially decouple energy generation from demand when power fluctuations are present [1]. Due to the technological developments in rechargeable batteries, interest has increased in energy storage systems based on electrochemical batteries.

Currently, a wide variety of battery technologies are available on the market, ranging from mature battery technologies such as lead-acid, nickel-cadmium (Ni-Cd), and nickel-metal hydride (Ni-MH) batteries to the recently developed technologies such as lithium-ion (Li-ion) batteries. Other battery technologies still under development are metal-air batteries. Amongst the existing battery technologies, Li-ion based batteries offer the most promising solution to the issues regarding power quality by providing ancillary services to the grid (i.e., the services that are required by the electrical grid to allow a continuous flow of electricity to ensure that supply meets demand).

In Figure 1.1, a comparison of different battery technologies in terms of their volumetric and gravimetric energy density is presented [2]. Li-ion based batteries have a high energy density, are lighter in weight, and have a longer life span; thus, they currently outperform other battery technologies.



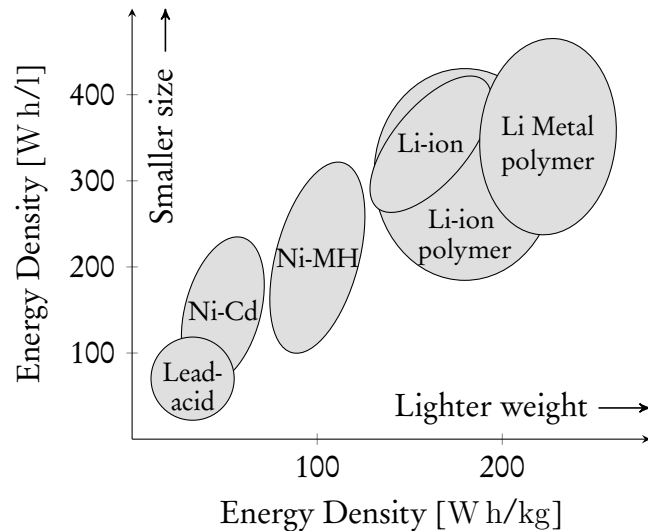


Figure 1.1: A comparison of different battery technologies. Replicated from [2].

Power electronics play a vital role in integrating renewable energy sources into the electrical grid. Likewise, they integrate renewable energy sources with small-scale stand-alone systems. Furthermore, on account of the fast evolution that power electronics have undergone over the years, the development of more efficient and grid-friendly converters has spiraled up, mainly due to two factors: the development of fast semiconductor switches that are capable of handling high power and the advent of real-time controllers that can implement complex control algorithms [3].

This thesis focuses on studying grid-tied Li-ion battery energy storage through a single-phase dc-to-ac converter, as shown in Figure 1.2. Also, a load connected on the ac side in place of the electrical grid can be used for the study, as will be discussed later. The converter provides an interface to draw or inject power into the electrical grid. Nonetheless, it is well-known that the instantaneous power generated on the ac side of a single-phase dc-to-ac converter pulsates at twice the grid frequency [4]. The pulsating power consists of both a dc component and a second-order harmonic component (i.e., the ripple power that oscillates at twice the grid frequency). As illustrated in Figure 1.2, the pulsating power  $p(t)$  is transferred from the ac side to the dc side.

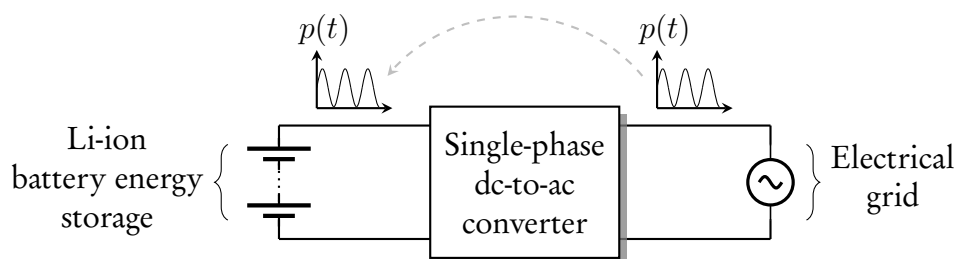


Figure 1.2: Block diagram illustrating the power transfer that occurs in a single-phase dc-to-ac converter from the ac side to the dc side, as indicated by the gray dashed arrow.

The pulsating power transferred onto the dc side inevitably generates second-order harmonic currents through the dc bus [4]. Unfortunately, this is a significant drawback associated with single-phase dc-to-ac converters. Considering that the system under investigation is powered by a battery, according to [5], the ripple current (i.e., the generated second-order harmonic

currents) can lead to immoderate chemical reactions during either the charging or discharging mode of the battery. As a result, the lifetime of the battery is significantly reduced. The acceptable ripple current passing through a battery should never exceed 10 % of the nominal battery current to ensure a longer lifespan [5].

The conventional method used to buffer the low-frequency ripple power to reduce the ripple current passing through the battery involves using a dc-link capacitor. By utilizing this method of ripple energy compensation, the dc-link capacitor is charged during the positive half cycle and discharged during the negative half cycle of the ripple power [6]. This way, the current passing through the battery is kept relatively constant since the battery power equals the dc component of the pulsating power, which was transferred onto the dc side. Figure 1.3 illustrates the conventional method of ripple energy compensation. The dc component and the second-order harmonic component of the pulsating power  $p(t)$  are denoted by  $P_o$  and  $p_r(t)$ , respectively.

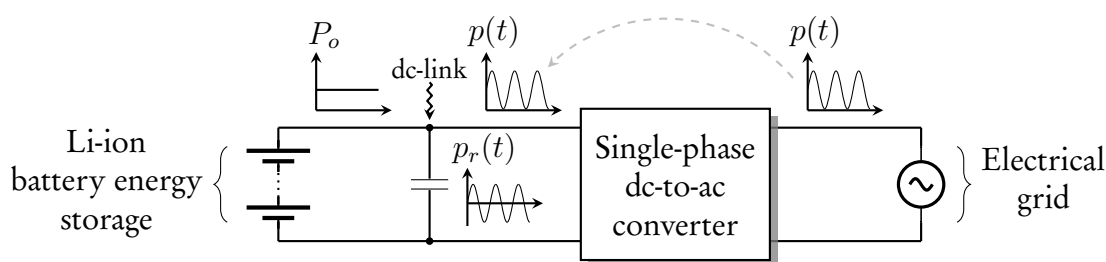


Figure 1.3: Block diagram showing the conventional method of ripple energy compensation, which utilizes a dc-link capacitor.

Usually, aluminium electrolytic capacitors are a common choice for use as dc-link capacitors. This is due to their availability in large capacitance values; hence, they can reduce the ripple current to an acceptable value. However, this type of capacitor is bulky, heavy, and has a short lifetime. It, therefore, contributes to an increase in the system's cost and volume significantly.

For applications that employ aluminium electrolytic capacitors, there exists a trade-off between reducing the ripple current passing through the battery and minimizing the total capacitance required. Figure 1.4 illustrates the trade-off. The aim is to optimize the trade-off point closer to the origin. In other words, to reduce the ripple current passing through the battery while utilizing a smaller capacitor.

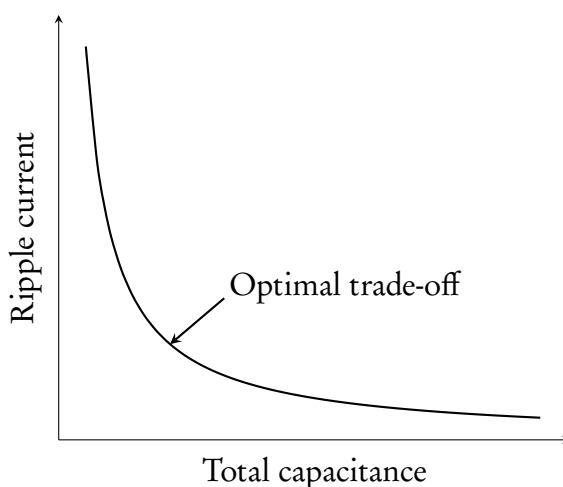


Figure 1.4: The trade-off between the ripple current and the total capacitance required.

Several approaches have been proposed in the literature to achieve the objective mentioned above. In particular, one approach employs an energy storage circuit connected on the dc side of the single-phase dc-to-ac converter [4, 7, 8, 9]. This approach is illustrated in Figure 1.5. In principle, the energy storage circuit is in the form of a dc-to-dc converter. The capacitor of the dc-to-dc converter, referred to as the *active capacitor*, is used for ripple energy compensation. The voltage across the active capacitor can vary over a wide range by employing this method of ripple energy compensation. As a result, more ripple energy is compensated for, and at the same time, a smaller capacitor is used.

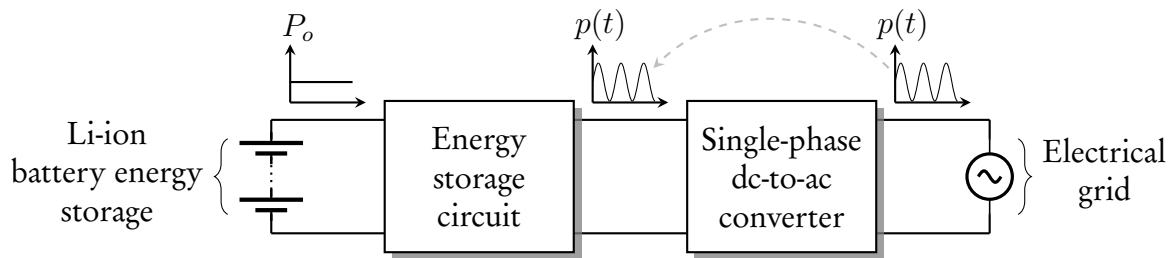


Figure 1.5: Block diagram showing the concept where an energy storage circuit is connected on the dc side of a single-phase dc-to-ac converter for ripple energy compensation.

Instead of using aluminium electrolytic (AE) capacitors, the metallized polypropylene film (MPPF) capacitors, also referred to as *thin-film* capacitors, can be used. They can handle higher ripple currents and have a longer lifespan. Figure 1.6 shows the relation between the ripple current rating and capacitance required for both low-ripple and high-ripple current applications when using either aluminium electrolytic or thin-film capacitors for ripple energy compensation [10].

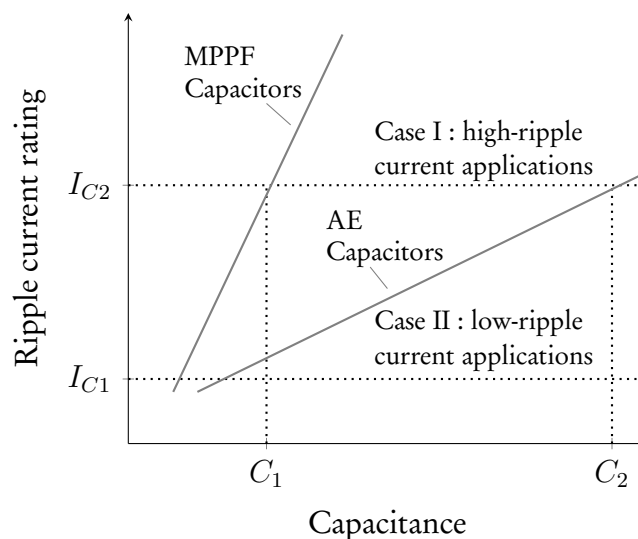


Figure 1.6: The required capacitance for low-ripple and high-ripple current applications. Replicated from [10].

When using an energy storage circuit for ripple energy compensation, a control problem is formulated. The ripple current needs to be actively diverted away from the battery to the active capacitor to keep the battery current relatively constant. To accomplish this, the current flowing through the energy storage circuit should closely track the ripple current. Hence, a control strategy that can successfully solve the current tracking problem is required.

Various control strategies proposed in the literature have significantly reduced the ripple current for different applications while utilizing a small capacitor. The controllers used are based mainly on the classical linear controllers; for example, the proportional-integral (PI) controller [4, 7, 8]. However, in order to achieve optimal performance, these controllers require laborious tuning of the gains. Once the optimal performance has been achieved, it only applies to a narrow operating range; beyond this range, their performance distinctly declines. Unfortunately, this poses challenges in both the theoretical and practical implementation of the controller.

Over the past decade, an interest in model predictive control (MPC) has tremendously increased in the field of power electronics due to the emergence of microprocessors and field-programmable gate arrays (FPGAs) with increased computational power [11, 12, 13, 14, 15]. Considering its design simplicity, fast dynamics, and systematic approach to solving control problems, particularly control problems involving reference tracking, MPC has become more attractive than other control strategies.

MPC is a model-based control strategy that employs the mathematical model of a system to forecast system behaviour over a finite prediction horizon. The control objectives are captured into a cost function, which is optimized to obtain the control action. To achieve good system performance often requires long prediction horizons, resulting in an MPC method known as *long-horizon* MPC. With long-horizon MPC, the computational burden of solving the underlying optimization problem grows exponentially. Fortunately, the optimization problem can be solved efficiently by utilizing optimization methods, namely, the branch-and-bound technique and the move blocking strategy, to yield a non-trivial prediction horizon.

## 1.2 Thesis objectives

Up to the present time, MPC has not been used in the literature to address the control problem associated with using an energy storage circuit for ripple energy compensation in a single-phase dc-to-ac converter. In this regard, this thesis aims to achieve the following objectives.

Firstly, to formulate a control strategy based on MPC. The formulated control strategy should allow the energy storage circuit to divert the generated ripple current on the dc bus away from the battery such that the ripple current passing through the battery does not exceed 10 % of the nominal battery current.

Secondly, the control strategy should incorporate optimization methods to efficiently solve the optimization problem while using long prediction horizons. In other words, solving the optimization problem of long horizons should not be computationally expensive. The use of long horizons should offer significant performance benefits over shorter horizons.

Thirdly, using both a grid-connected system and a stand-alone system, the effectiveness of the proposed control strategy should be verified via MATLAB® simulations during steady-state operation and transients. It is noteworthy that a stand-alone system also generates second-order harmonic currents through the dc bus. Furthermore, to evaluate the performance benefits of long horizons, the target harmonic that needs to be reduced should be analyzed for different prediction horizons. This work was published in [16].

Lastly, on a low-cost FPGA, the MPC-based control strategy with long horizons should be implemented practically. It should be implemented carefully without sacrificing the FPGA's available resources since FPGAs only have a finite amount of resources. Through a hardware-in-the-loop (HiL) simulation of a stand-alone system on the FPGA, the practical feasibility of the controller design should be demonstrated by comparing the HiL simulation results with those obtained from MATLAB® simulations.

A summary of the main contributions of this thesis are as follows:

- The formulation of a control strategy based on long-horizon MPC for an energy storage circuit to reduce the ripple current passing through the battery such that it does not exceed 10 % of the nominal battery current.
- The verification of the formulated control strategy through MATLAB® simulations of a grid-connected system and a stand-alone system.
- The practical implementation of the formulated control strategy and validation of the controller design via a HiL simulation of the stand-alone system on a low-cost FPGA.

### 1.3 Thesis outline

This thesis constitutes six chapters. The content of the chapters is briefly discussed.

**Chapter 1: Introduction** provides the background information relevant to this thesis. The major drawback associated with a grid-tied single-phase dc-to-ac converter powered by a Li-ion battery energy storage system is explained briefly. The drawback relates to the transfer of pulsating power from the ac side to the dc side, and thenceforth, generating second-order harmonic currents through the dc bus. The conventional method used to mitigate second-order harmonic currents and its corresponding disadvantages is mentioned. Furthermore, the concept of using an energy storage circuit for ripple energy compensation and the control schemes currently used to address the associated control problem are stated. A brief description of MPC is presented as the proposed control strategy when utilizing an energy storage circuit.

The objectives of this thesis, which are mainly based on the proposed control strategy and its implementation, are presented. The primary objective is that the ripple current passing through the battery should be limited to 10 % of the nominal battery current.

**Chapter 2: Theoretical background** gives a foundation of the theory applied in this thesis. Initially, the chapter presents the topology of a single-phase dc-to-ac converter. The modulation techniques used to control the semiconductor switches of the converter are discussed in detail. A detailed analysis of the generation of the second-order harmonic currents in a single-phase dc-to-ac converter is given. The effect of the ripple current on the battery energy storage system is explained. Moreover, the methods used for ripple energy compensation are further discussed. Focus is given to the method that employs a dc-to-dc boost converter as an energy storage circuit. The control strategies used historically to address the associated control problem are briefly reviewed.

MPC is introduced formally. Its advantages, fundamental components, and principles are discussed. This includes the internal dynamic model of the system, constraints, formulation of the optimal control problem, and the receding horizon policy. The exhaustive search method for solving the optimization problem is explained. The chapter concludes by introducing two optimization methods: the branch-and-bound technique and the move blocking strategy, which are used to reduce the computational burden of using long-horizon MPC effectively. Throughout the chapter, suitable examples are given.

**Chapter 3: Model predictive control of a dc-to-dc boost converter** discusses the modelling and the formulation of the optimal control problem for a dc-to-dc boost converter. A discussion on the difficulty encountered when directly controlling the capacitor voltage, due to the non-minimum phase nature of the system, is given. The importance of using long prediction horizons to address the associated problem is discussed with a relevant example. Moreover, the move blocking strategy is suggested as an optimization method that maintains a sufficiently long prediction horizon without a subsequent increase in the computational burden. A non-recursive MPC algorithm that implements the branch-and-bound technique to further reduce the computations required and subsequently improve the performance of the system is formulated. The algorithm also incorporates the move blocking strategy. These two optimization methods allow the implementation of long-horizon direct MPC in a time-efficient manner on an FPGA.

The use of a band-pass filter to generate an accurate reference ripple current for the controller is explained. An accurate reference to the inductor current of the boost converter ensures the possibility of ripple energy compensation in the single-phase dc-to-ac converter.

**Chapter 4: Performance evaluation of long prediction horizons** presents the simulation results of a grid-connected and a stand-alone system to verify the effectiveness of long-horizon MPC. The simulation waveforms shown are for 10-step predictions, which is the longest horizon considered. The capacitance requirements for ripple energy compensation for both systems are significantly reduced by over 95 %.

It is shown that the ripple current flowing through the battery is reduced to approximately 2.4 % and 2.8 % of the nominal battery current, for the grid-connected system and the stand-alone system, respectively. The boost converter operated at a switching frequency of about 16 kHz. The performance benefits of long horizons are examined for selected horizons that can attain steady-state operation, i.e., prediction steps greater than two. The response of the controller during transients is investigated by applying step changes in the load current of the stand-alone system.

**Chapter 5: FPGA implementation** discusses the implementation of a HiL simulation of the stand-alone system, the pulse-width modulator and the model predictive controller on an FPGA. Firstly, preliminary information on FPGAs and their working principle is presented. The notion of computational delay encountered when implementing the model predictive controller practically is discussed and the method used to compensate for the delay presented.

A HiL simulation of the stand-alone system and the model predictive controller with long horizons is conducted within the FPGA. The VHDL implementation of the controller is verified through a comparison of the HiL simulations with the MATLAB® simulations for 3-step to 7-step predictions. It is shown that for the longest horizon considered on the FPGA with 7-step predictions, the ripple current passing through the battery is reduced to 3.2 % of the nominal battery current with the practical controller on the FPGA.

**Chapter 6: Conclusions** concludes this thesis. The key results obtained from relevant chapters are summarized. Recommendations for future research based on this thesis are given.

# Chapter 2

## Theoretical background

This chapter is subdivided into two parts. The fundamental principles and basic concepts required for this thesis are summarized. The first part (Part I) describes the problem statement in detail, and the second part (Part II) formally introduces model predictive control.

---

### Part I

---

This part of the chapter starts by facilitating the selection of suitable semiconductor devices for use in the converter. An introduction to the basic topology of the single-phase dc-to-ac converter is given. The modulation technique used to control the semiconductor switches of the converter is discussed in detail. Furthermore, a detailed analysis of the pulsating power generated by the converter is presented. The effect that the ripple current has on the battery energy storage system and the methods employed to compensate for the ripple energy are discussed, with a focus on ripple energy compensation using an energy storage circuit. The part concludes with a brief discussion of classical control schemes used to address the associated control problem of utilizing an energy storage circuit for ripple energy compensation.

## 2.1 The single-phase dc-to-ac converter

### 2.1.1 Power semiconductor devices

Power semiconductor devices form the core of most power electronic applications. Therefore, before formally introducing the topology of the single-phase dc-to-ac converter, it is essential to select the semiconductor devices that best suit the implementation of the converter.

There exist a vast range of semiconductor devices on the market. The most common ones that are commercially available include diodes, thyristors, bipolar junction transistors (BJTs), metal-oxide-semiconductor field-effect transistors (MOSFETs), and insulated-gate bipolar transistors (IGBTs). Out of the devices mentioned above, the MOSFET and the IGBT have gained popularity in recent years due to their improved operating principles, their specifications, and performance [17].

The choice between MOSFETs and IGBTs is application based. More specifically, IGBTs are more suitable for medium and high voltage applications (higher than 1 kV) [17]. On the other hand, MOSFETs have the advantage of providing good efficiency at lower voltages (below 250 V). In terms of the switching characteristics, MOSFETs are capable of operating at higher switching frequencies that are beyond the 20 kHz spectrum, conversely, IGBTs perform better at lower switching frequencies [17]. At higher switching frequencies, IGBTs tend to exhibit higher switching losses. Moreover, MOSFETs have a low on-state resistance, and as a result, they have lower conduction losses.

Due to the specifications mentioned above, MOSFETs are chosen as the semiconductor switches instead of IGBTs. They are well suitable for the required application in this thesis.

### 2.1.2 Introduction to topology

The basic structure of a full-bridge single-phase dc-to-ac converter is shown in Figure 2.1.

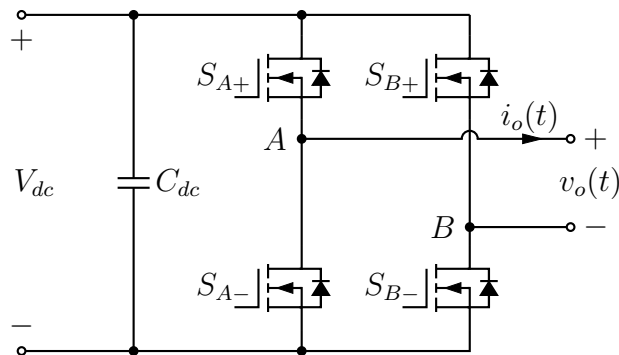


Figure 2.1: A full-bridge single-phase dc-to-ac converter with the semiconductor switches as MOSFETs along with their associated built-in anti-parallel diodes.

The full-bridge single-phase dc-to-ac converter consists of two phase arms of the same type.<sup>1</sup> The phase arms are connected to a common dc bus [18]. The first phase arm, *phase arm A*, has two semiconductor switches  $S_{A+}$  and  $S_{A-}$ . The second phase arm, *phase arm B*, has two semiconductor switches  $S_{B+}$  and  $S_{B-}$ . Each semiconductor switch comprises a built-in anti-parallel diode that allows the conduction of current in both directions. On the dc side, a dc-link capacitor  $C_{dc}$  is connected in parallel with a dc supply voltage, denoted by  $V_{dc}$ . At the output terminal on the ac side, a voltage, denoted by  $v_o(t)$ , is produced by controlling the semiconductor switches of the two phase arms.

It is important to note that the two switches in a phase arm should not be switched on at the same time to avoid short-circuiting the dc supply voltage [19]. In practice, a small-time delay, referred to as the *blanking time*, should be included between the switching transitions of the two switches in a phase arm. In this thesis, the blanking time is not taken into account for simplification; thus, the switching of the semiconductor switches is assumed to be ideal.

In the next section, the modulation technique used to control the semiconductor switches of the single-phase dc-to-ac converter will be discussed in detail.

<sup>1</sup>From here on, the “full-bridge single-phase dc-to-ac converter” will only be referred to as the single-phase dc-to-ac converter.



### 2.1.3 Carrier-based pulse-width modulation

In power electronics, the most common modulation technique is pulse-width modulation (PWM). The general idea behind PWM involves translating a real-valued reference signal into a discrete-valued switching signal. The switching signal is used as gating signals for the converter semiconductor switches. Generally, the discrete-valued switching signal is a pulse with a fixed amplitude, variable-width, and modulates at a constant frequency, as the name suggests.

*Carrier-based* pulse-width modulation (CB-PWM) for a single-phase dc-to-ac converter is achieved by comparing a reference signal  $u_{ref}(t)$  with a carrier signal  $u_{car}(t)$ . The reference signal is a sinusoidal waveform with a frequency  $f_1$ , where  $f_1$  represents the fundamental frequency. Note that the frequency of the reference signal defines the frequency of the output voltage  $v_o(t)$ . The magnitude of the reference signal is the so-called *modulation index*,

$$m_a = \frac{\hat{u}_{ref}}{\hat{u}_{car}}, \quad (2.1)$$

with  $\hat{u}_{ref}$  and  $\hat{u}_{car}$  as the amplitudes of the reference signal and the carrier signal, respectively. It is important to note that the modulation index is limited to  $m_a \in [0, 1]$ . The carrier signal is a triangular waveform with a frequency  $f_c$  and an amplitude that ranges from  $-1$  to  $1$ . The frequency of the carrier signal is proportional to the switching frequency of the converter. Note that the frequency of the carrier signal is much higher than the fundamental frequency, i.e.,  $f_c \gg f_1$ .

This modulation strategy can employ a sampling technique referred to as *natural sampling*. It works as follows: the switching instants occur at the intersection of the reference signal and the carrier signal. As a result, a pulse is generated, which is the pulse-width modulated signal. The pulse is applied to the semiconductor switches as a gating signal, and thus, an output voltage is produced at the ac side terminal. The produced output voltage

$$v_o(t) = V_{dc}u_{pwm}(t), \quad (2.2)$$

where  $u_{pwm}(t)$  denotes the switching function. The switching function is related to the switch positions of the generated pulse-width modulated signals, and it depends on the switching strategy employed. Two CB-PWM switching strategies for single-phase dc-to-ac converters can be used to produce the switching function. The switching strategies are described as:

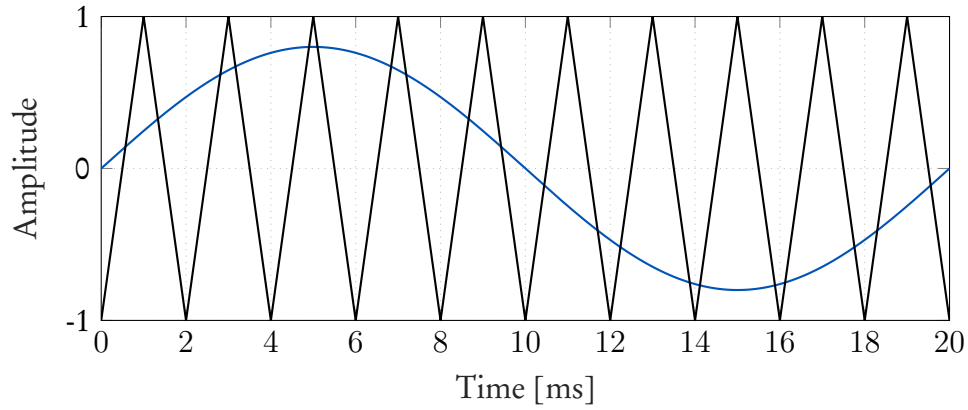
1. PWM with bipolar switching,
2. PWM with unipolar switching.

#### Bipolar switching strategy

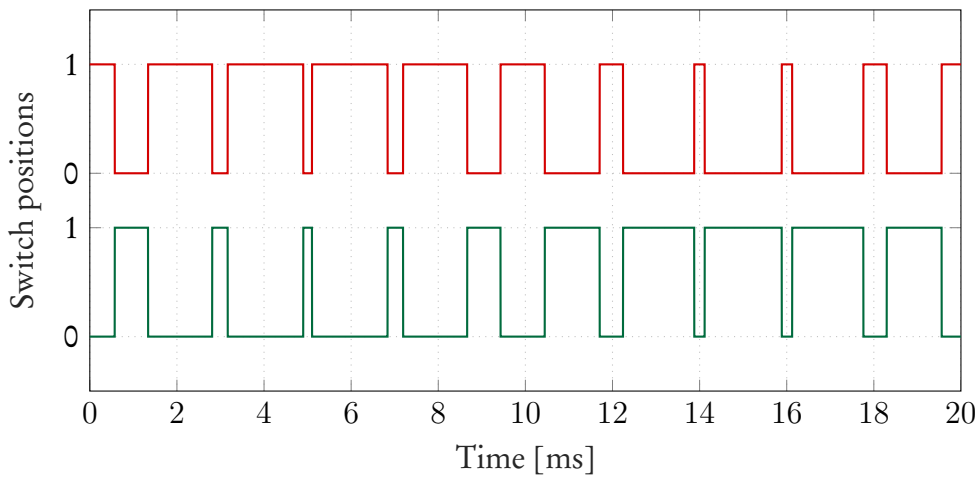
By using this switching strategy, the semiconductor switches of the converter are switched as two switch pairs; switches  $S_{A+}$  and  $S_{B-}$  switch together; switches  $S_{A-}$  and  $S_{B+}$  switch together [19]. In other words, the two switches that are diagonally opposite to each other are switched on and off simultaneously. The switching strategy is illustrated in Figure 2.2. A sinusoidal reference signal

$$u_{ref}(t) = m_a \sin(\omega_1 t), \quad (2.3)$$

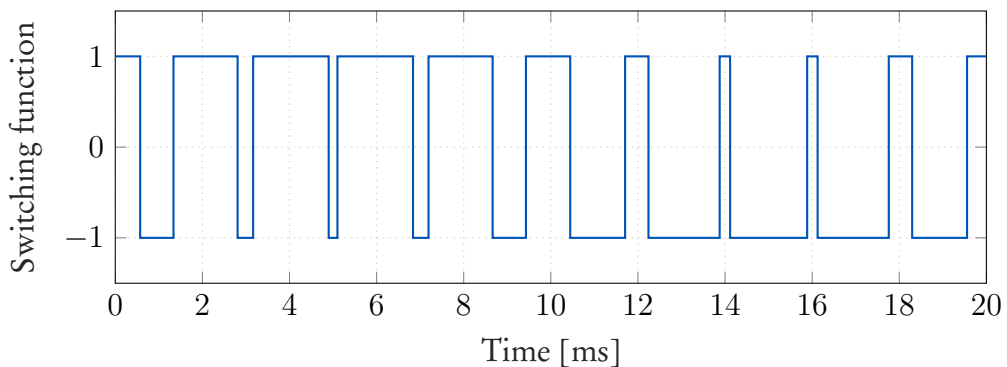
with  $\omega_1 = 2\pi f_1$  as the angular frequency and a triangular carrier signal are used to achieve modulation with natural sampling.



(a) Illustration of  $u_{ref}(t)$  the reference signal (blue sinusoidal waveform) and  $u_{car}(t)$  the carrier signal (black triangular waveform) for CB-PWM with bipolar switching.



(b) Illustration of the switch positions corresponding to the modulation of the reference signal with the carrier signal. The top signal (red pulse) represents  $u_{sw}(t)$  and the bottom signal (green pulse) represents  $\overline{u_{sw}}(t)$ .



(c) Illustration of the switching function  $u_{pwm}(t)$  resulting from CB-PWM with bipolar switching.

Figure 2.2: Illustrative example of pulse-width modulation with bipolar switching. The amplitude of the reference signal is the modulation index, given a value  $m_a = 0.8$ . The fundamental frequency  $f_1 = 50$  Hz. The carrier signal has a frequency  $f_c = 500$  Hz.

Figure 2.2a depicts natural sampling. The reference signal is modulated as follows:

- if  $u_{ref}(t) > u_{car}(t)$ , then  $u_{sw} = 1$
- if  $u_{ref}(t) < u_{car}(t)$ , then  $u_{sw} = 0$ ,

where  $u_{sw} \in \{0, 1\}$  represents the switch positions of the pulse-width modulated signal as shown in Figure 2.2b. The switching signal  $u_{sw}(t)$  is applied to the semiconductor switch pair,  $S_{A+}$  and  $S_{B-}$ . While the switching signal  $\overline{u_{sw}}(t)$ , the inverse of  $u_{sw}(t)$ , is applied to the semiconductor switch pair,  $S_{A-}$  and  $S_{B+}$ .

The switching function resulting from PWM with bipolar switching is given by

$$u_{pwm}(t) = u_{sw}(t) - \overline{u_{sw}}(t), \quad (2.4)$$

where  $u_{pwm} \in \{-1, 1\}$  as illustrated in Figure 2.2c. By using (2.2), the produced output voltage waveform jumps between  $+V_{dc}$  and  $-V_{dc}$  [19]. Table 2.1 gives a summary of the switching function, the semiconductor switch states when active (1) and inactive (0), and the output voltage generated from controlling a single-phase dc-to-ac converter using PWM with bipolar switching.

Table 2.1: Switching states and the resulting output voltage for single-phase dc-to-ac converter.

Switching function $u_{pwm}$	Active/Inactive switch				Output voltage $v_o$
	$S_{A+}$	$S_{A-}$	$S_{B+}$	$S_{B-}$	
1	1	0	0	1	$V_{dc}$
-1	0	1	1	0	$-V_{dc}$

Although CB-PWM with bipolar switching is easy to implement, there is a major disadvantage of this switching strategy. The large voltage variation of  $2V_{dc}$  at the voltage output terminal caused by the jump between  $V_{dc}$  and  $-V_{dc}$  results in a significant amount of electromagnetic interference (EMI) and high total harmonic distortion (THD) [20]. As a consequence, the output voltage  $v_o(t)$  and current  $i_o(t)$  have high ripple content.

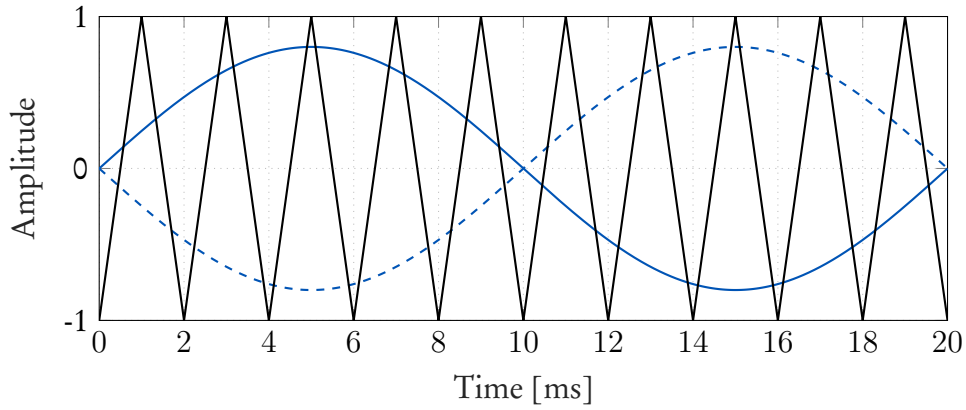
### Unipolar switching strategy

With unipolar switching, the semiconductor switches of each phase arm in the single-phase dc-to-ac converter are switched independently of the other phase arm [19]. The switching strategy is illustrated in Figure 2.3. The phase arms are modulated by two reference signals that are out of phase with each other by  $180^\circ$ , as defined by

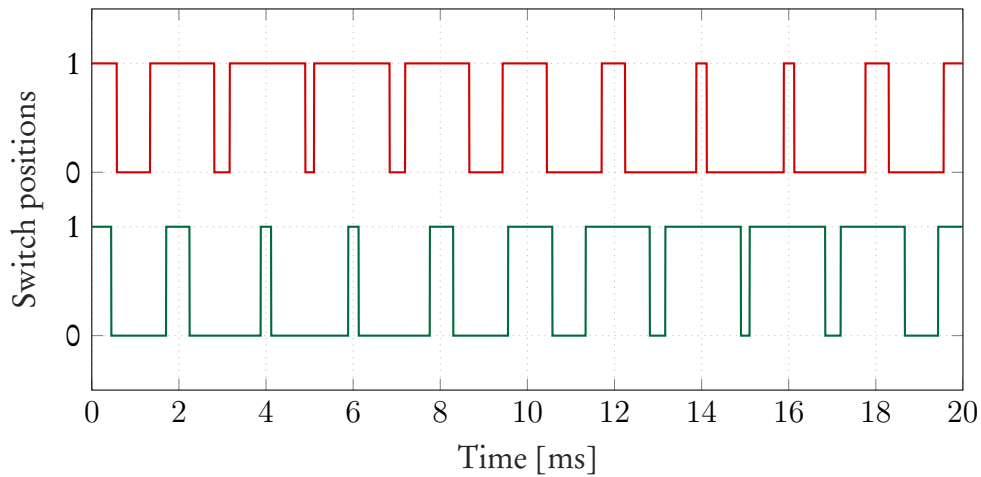
$$\mathbf{u}_{ref}(t) = m_a \begin{bmatrix} \sin(\omega_1 t) \\ \sin(\omega_1 t - \pi) \end{bmatrix}. \quad (2.5)$$

Both phase arms use a common triangular carrier signal to achieve modulation.<sup>2</sup> Similar to bipolar switching, natural sampling is used to modulate the reference signals.

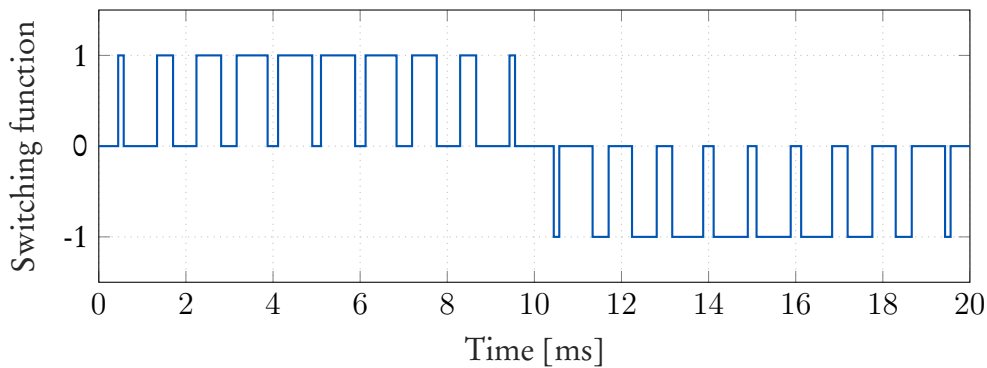
<sup>2</sup>Note that using a common carrier signal and two sinusoidal reference signals is not mandatory to achieve modulation with the unipolar switching strategy. Other variations have been proposed in the literature that uses different combinations of the carrier and reference signals that offer certain advantages for specific applications.



(a) Illustration of  $u_{ref,A}(t)$  (blue solid sinusoidal waveform) and  $u_{ref,B}(t)$  (blue dashed sinusoidal waveform), the reference signals, and  $u_{car}(t)$  the carrier signal (black triangular waveform) for CB-PWM with unipolar switching.



(b) Illustration of the switch positions corresponding to the modulation of reference signals with the carrier signal. The top signal (red pulse) represents  $u_{sw,A}(t)$  and the bottom signal (green pulse) represents  $u_{sw,B}(t)$ .



(c) Illustration of the switching function  $u_{pwm}(t)$  resulting from CB-PWM with unipolar switching.

Figure 2.3: Illustrative example of pulse-width modulation with unipolar switching. The amplitude of the reference signals is the modulation index, given the value  $m_a = 0.8$ . The fundamental frequency  $f_1 = 50$  Hz. The carrier signal has a frequency  $f_c = 500$  Hz.

Figure 2.3a illustrates natural sampling. Let  $u_{ref,A}(t)$  and  $u_{ref,B}(t)$  denote the first and the second entries of the reference signal vector  $\mathbf{u}_{ref}(t)$  in (2.5), respectively. The reference signal for phase arm  $A$  is modulated as follows:

- if  $u_{ref,A}(t) > u_{car}(t)$ , then  $u_{sw,A} = 1$
- if  $u_{ref,A}(t) < u_{car}(t)$ , then  $u_{sw,A} = 0$ ,

and the reference signal for phase arm  $B$  is modulated as follows:

- if  $u_{ref,B}(t) > u_{car}(t)$ , then  $u_{sw,B} = 1$
- if  $u_{ref,B}(t) < u_{car}(t)$ , then  $u_{sw,B} = 0$ ,

where  $u_{sw,A} \in \{0, 1\}$  and  $u_{sw,B} \in \{0, 1\}$  represent the switch positions of the pulse-width modulated signals for phase arm  $A$  and phase arm  $B$ , respectively, as shown in Figure 2.3b.

**Phase arm  $A$ :** The switching positions  $u_{sw,A}(t)$  are applied to the semiconductor switch  $S_{A+}$ . While the switching positions  $\overline{u_{sw,A}}(t)$ , the inverse of switch positions  $u_{sw,A}(t)$ , are applied to the semiconductor switch  $S_{A-}$ .

**Phase arm  $B$ :** The switching positions  $u_{sw,B}(t)$  are applied to the semiconductor switch  $S_{B+}$ . While the switching positions  $\overline{u_{sw,B}}(t)$ , the inverse of switch positions  $u_{sw,B}(t)$ , are applied to the semiconductor switch  $S_{B-}$ .

This modulation technique is also known as *double-edge naturally sampled modulation* [18]. It results in a switching function

$$u_{pwm}(t) = u_{sw,A}(t) - u_{sw,B}(t), \quad (2.6)$$

where  $u_{pwm} \in \{-1, 0, 1\}$ , as shown in Figure 2.3c. By using (2.2), the produced output voltage waveform  $v_o(t)$  has three voltage levels, that is,  $+V_{dc}$ ,  $0$  V, and  $-V_{dc}$ .

The disadvantage of unipolar switching in comparison with bipolar switching is common-mode voltage [21]. The common-mode voltage variations occur at the midpoints of the two phase arms. They cause a large ground leakage current in grid-connected PV applications. Nevertheless, this switching strategy generates less EMI and a reduced THD than bipolar switching due to a voltage variation of only  $V_{dc}$  at the voltage output terminal. Thus, only PWM with unipolar switching will be considered.

Consider a single-phase dc-to-ac converter with the semiconductor switches controlled by PWM unipolar switching. Figure 2.4 shows the current paths when the output current  $i_o(t)$  is positive, for different switch positions. Figure 2.5 shows the current paths when  $i_o(t)$  is negative, for different switch positions. The current paths are highlighted in black. Here, an assumption is made that the drain-to-source voltage  $V_{DS}$  of the MOSFETs is greater than the on-state voltage  $V_{on}$  of the anti-parallel diode. Thus, current will only flow through the resistive region of the MOSFET instead of the anti-parallel diode.

When both the upper semiconductor switches ( $S_{A+}$  and  $S_{B+}$ ) are on at the same time, or when both the lower semiconductor switches ( $S_{A-}$  and  $S_{B-}$ ) are on at the same time, that's when an output voltage of  $0$  V is produced. As a result, four different switching states are possible even though two of them are redundant. Table 2.2 summarizes the switching function, the corresponding semiconductor switching states, and the output voltage generated from controlling a single-phase dc-to-ac converter using PWM with unipolar switching.

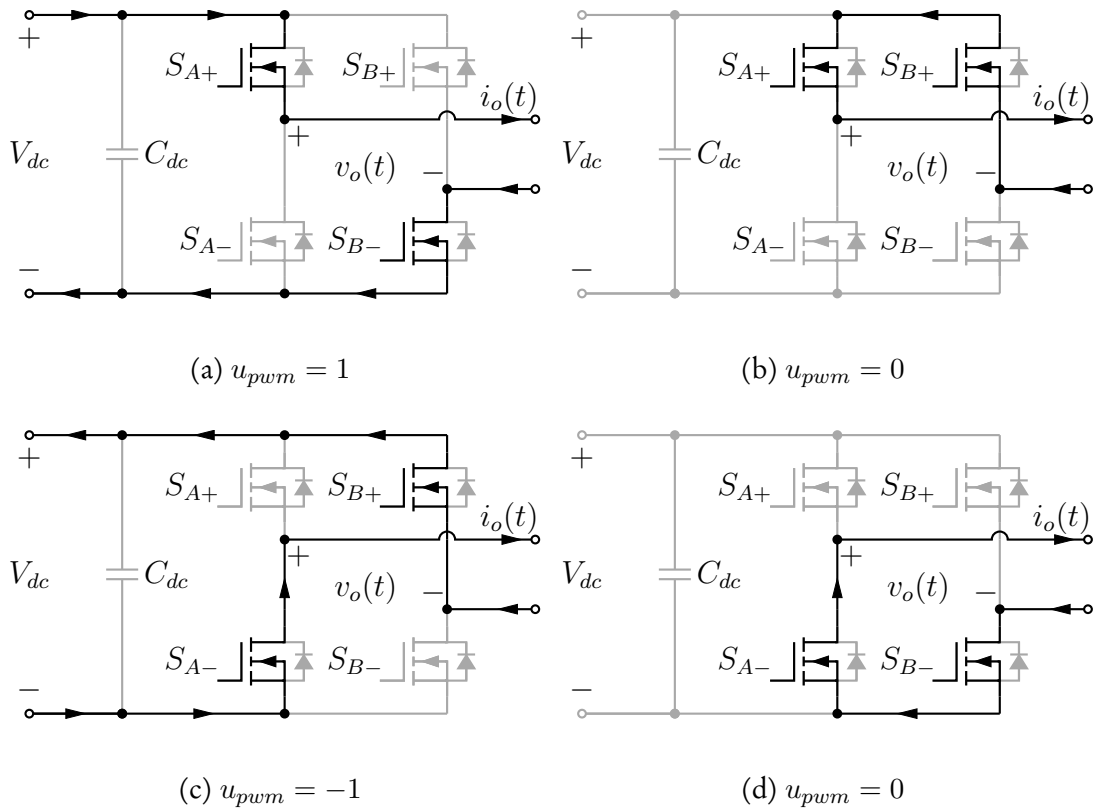


Figure 2.4: Positive current paths through the single-phase dc-to-ac converter switches.

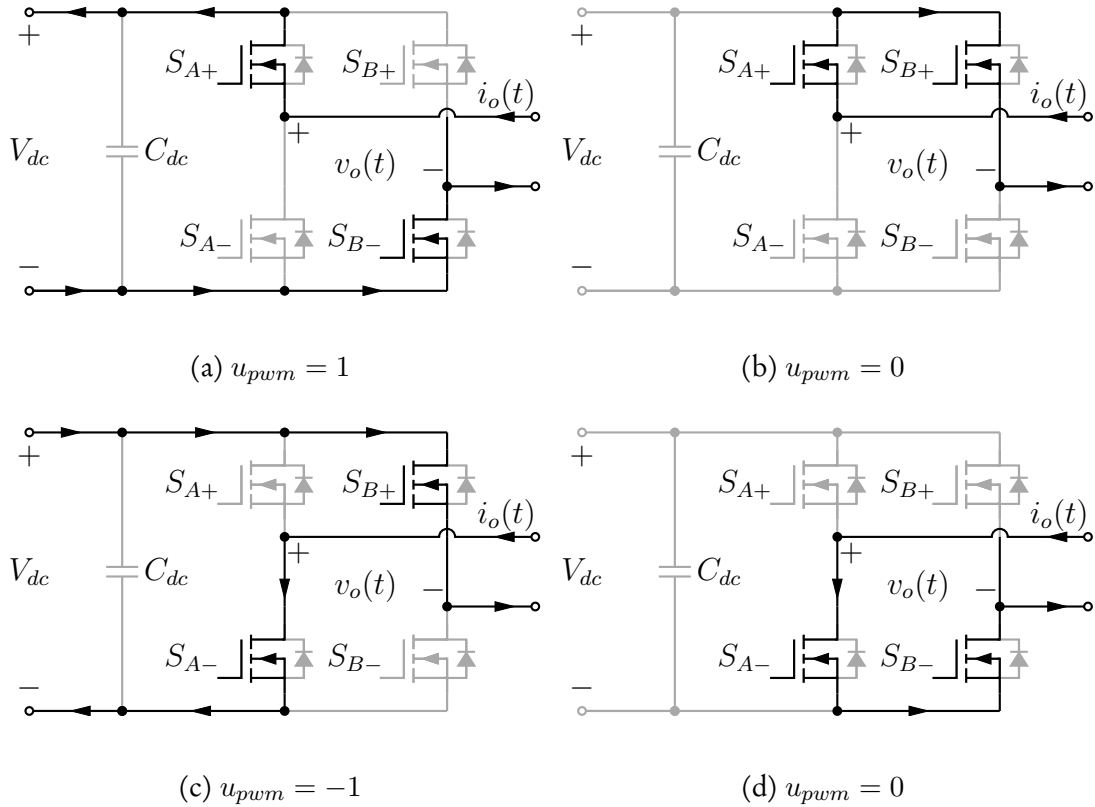


Figure 2.5: Negative current paths through the single-phase dc-to-ac converter switches.

Table 2.2: Switching states and the resulting output voltage for single-phase dc-to-ac converter.

Switching function $u_{pwm}$	Active/Inactive switch				Output voltage $v_o$
	$S_{A+}$	$S_{A-}$	$S_{B+}$	$S_{B-}$	
1	1	0	0	1	$V_{dc}$
0	1	0	1	0	0
-1	0	1	1	0	$-V_{dc}$
0	0	1	0	1	0

### Harmonic analysis

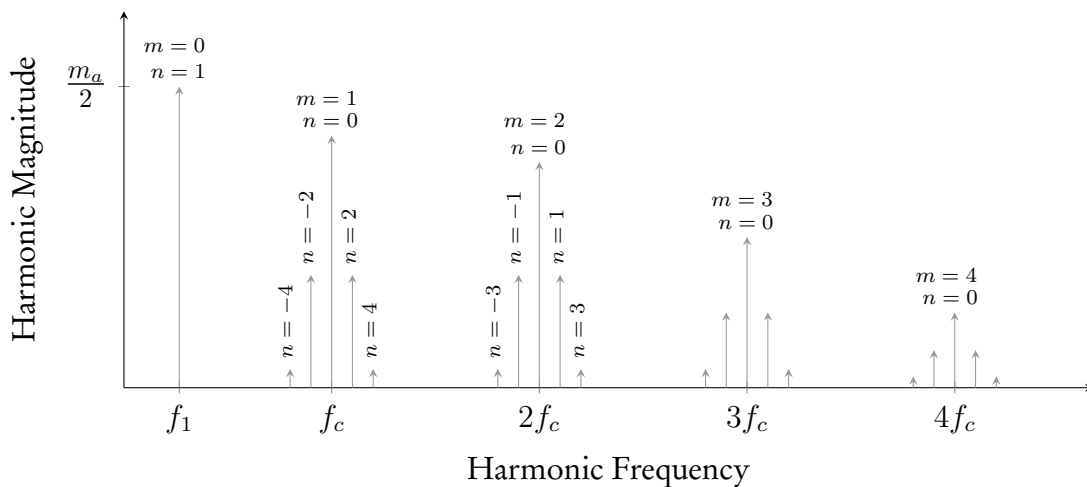
The weighted infinite sum of sinusoidal signals, known as the *Fourier series*, can be used to represent any periodic signal. The process of computing and analyzing the Fourier series is known as *harmonic analysis*.

Consider CB-PWM with unipolar switching for a single-phase dc-to-ac converter, as discussed in the previous section. The resulting switching transitions of the pulse-width modulated signal(s) are dependent on the reference signal and carrier signal. As already mentioned, the signals used in the modulation process have different frequencies; the carrier signal with a frequency  $f_c$  and the two reference signals with a fundamental frequency  $f_1$ . The formulated Fourier series incorporates both the reference and carrier signal to account for both frequencies. In other words, the formulated Fourier series is a function of the carrier and reference signals, thus, the Fourier coefficients of the series employ two sets of integrals to form a *double Fourier series integral*. The interested reader is referred to [18], for a more detailed analysis of the mathematical derivations of the double Fourier integral. This section is solely dedicated to analyzing the harmonics resulting from PWM with unipolar switching.

Due to the switching nature of PWM, it is well-known that the resulting modulated signal for each phase arm contains harmonics that are located at the frequencies

$$f_{mn} = mf_c + nf_1, \quad (2.7)$$

where  $m \in \mathbb{N}$  is the carrier index variable and  $n \in \mathbb{Z}$  is the baseband index variable. From (2.7)  $mf_c$  represents the integer multiple of the carrier frequency and  $nf_1$  represents the integer multiple of the fundamental frequency. Figure 2.6 illustrates the harmonics resulting from  $u_{sw,A}(t)$ , used to control the semiconductor switches of phase arm  $A$ .

Figure 2.6: Illustration of the harmonic spectrum of the switch positions  $u_{sw,A}(t)$ .

The harmonics occur in groups. When  $m = 0$  the harmonic frequencies are only defined by  $n$ ; this group of harmonics is referred to as the baseband harmonic components [18]. The harmonics that are only defined by  $m$ , when  $n = 0$  are defined as the carrier harmonic components [18]. Below is a summary explaining the harmonics shown in Figure 2.6.

---

The first baseband harmonic component is the fundamental component:

- $f_{0n} = f_1$ , with

$$m = 0 \text{ and } n = 1, \quad (2.8)$$

which defines the fundamental frequency  $f_1$ , i.e., the frequency of the reference signal.

---

The carrier multiple harmonics:

- $f_{m0} = mf_c$ , with

$$m \in \{1, 2, 3, \dots\} \text{ and } n = 0, \quad (2.9)$$

define the high-frequency components that correspond to the carrier signal.

---

Around the carrier harmonic components, are harmonics that exist as groups referred to as sideband harmonics:

- $f_{mn} = mf_c + nf_1$ , with

$$\begin{cases} m \in \{1, 3, 5, \dots\} \text{ and } n \in \{\pm 2, \pm 4, \pm 6, \dots\} \\ m \in \{2, 4, 6, \dots\} \text{ and } n \in \{\pm 1, \pm 3, \pm 5, \dots\}. \end{cases} \quad (2.10)$$


---

It should be noted that only even harmonic sideband components exist around odd carrier harmonics and only odd sideband harmonics exist around even carrier harmonics. [18]. This characteristic is a result of double-edge naturally sampled PWM.

The harmonics resulting from the pulse-width modulated signal for phase arm  $B$  are similar to that of phase arm  $A$ . The only difference is that the fundamental component and the even carrier harmonics, along with their associated sideband harmonics, are at a phase angle of  $180^\circ$  for phase arm  $B$ . However, the odd carrier harmonics and the even sideband harmonics are the same as those for phase arm  $A$ .

For this reason, when the modulated signals of the two phase arms,  $u_{sw,A}(t)$  and  $u_{sw,B}(t)$ , are subtracted from each other to generate the switching function  $u_{pwm}(t)$ , as shown in (2.6), the odd carrier harmonics along with their associated sideband harmonics are cancelled completely by virtue of unipolar modulation. For a detailed analysis of the mathematical derivations regarding the cancellation process, the reader is referred to [18].

As a result of PWM with unipolar switching, the harmonic spectrum of the switching function  $u_{pwm}(t)$  only consists of the fundamental component, the even carrier harmonics, and the odd sideband harmonic terms. Figure 2.7 illustrates the harmonic spectrum.



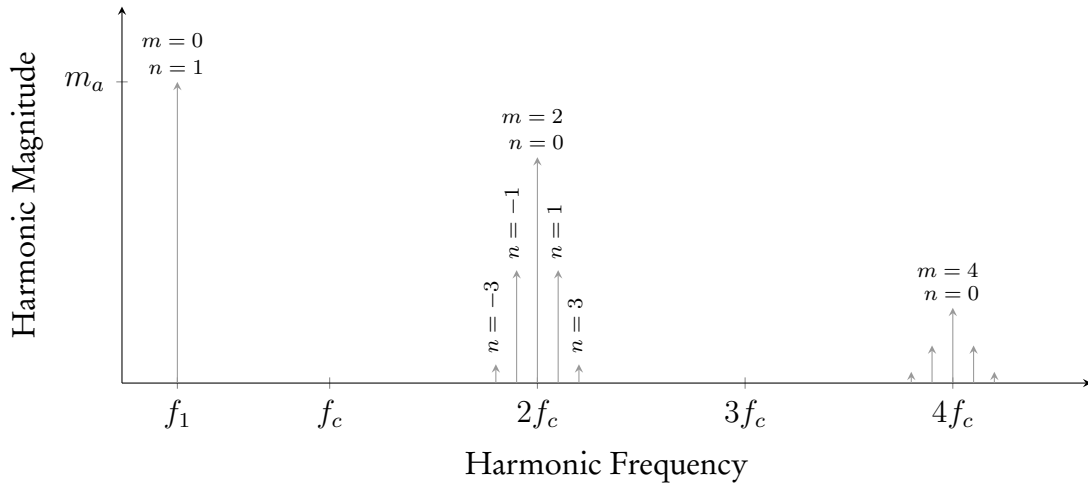


Figure 2.7: Illustration of the harmonic spectrum of the switching function  $u_{pwm}(t)$ .

The cancellation of the harmonics is a major advantage of unipolar switching. The switching function has a frequency twice that of the carrier signal, which implies that the switching frequency of the single-phase dc-to-ac converter is doubled. As a result, the output voltage  $v_o(t)$  has a reduced THD; thus, improving the power quality of the converter.

### 2.1.4 Ripple power in a single-phase dc-to-ac converter

Consider the single-phase dc-to-ac converter introduced in Section 2.1.2. The converter is connected to an ac grid with voltage and current given by

$$v_{ac}(t) = V_{ac} \sin(\omega_1 t) \quad (2.11a)$$

$$i_{ac}(t) = I_{ac} \sin(\omega_1 t - \phi), \quad (2.11b)$$

respectively. The amplitude of the grid voltage is denoted by  $V_{ac}$  and the amplitude of the grid current is denoted by  $I_{ac}$ . The phase angle between the voltage and current is denoted by  $\phi$ . Figure 2.8 shows the grid-connected single-phase dc-to-ac converter that is powered by a battery on the dc side.

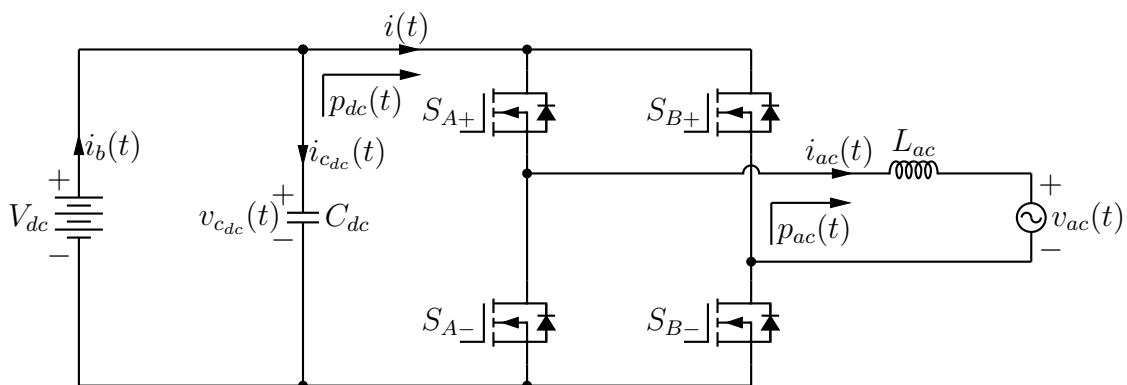


Figure 2.8: Grid-connected single-phase dc-to-ac converter with a dc-link capacitor.

The instantaneous power injected into the ac grid

$$\begin{aligned} p_{ac}(t) &= v_{ac}(t)i_{ac}(t) \\ &= \frac{V_{ac}I_{ac}}{2} \cos(\phi) - \frac{V_{ac}I_{ac}}{2} \cos(2\omega_1 t - \phi), \end{aligned} \quad (2.12)$$

consists of two terms: the *dc component* and the undesirable *second-order harmonic component*. The dc component is denoted by

$$P_o = \frac{V_{ac}I_{ac}}{2} \cos(\phi), \quad (2.13)$$

and the second-order harmonic component, also referred to as the *ripple power* is denoted by

$$p_r(t) = -\frac{V_{ac}I_{ac}}{2} \cos(2\omega_1 t - \phi). \quad (2.14)$$

The ripple power pulsates at twice the grid frequency, i.e., at a frequency that is two times the fundamental frequency  $f_1$ . From here on, the frequency of the ripple power will be indicated by  $f_r = 2f_1$ . Figure 2.9 illustrates the ripple power.

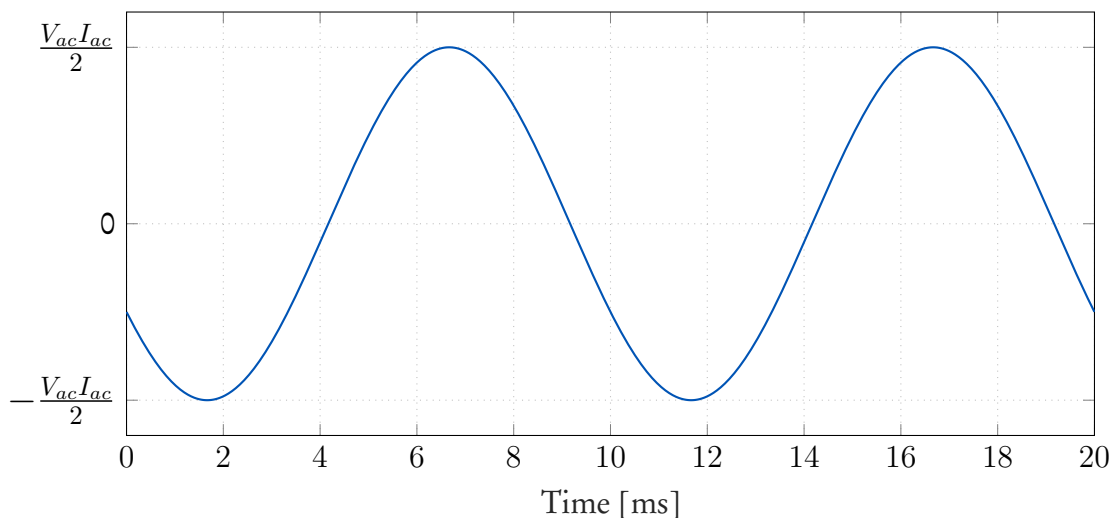


Figure 2.9: Example illustrating the ripple power  $p_r(t)$ . The ripple power is oscillating at a frequency  $f_r = 100$  Hz, since the fundamental frequency  $f_1 = 50$  Hz. The phase angle is set to  $\phi = \frac{\pi}{3}$  [rad] for illustrative purposes.

According to the power balance relationship, if an ideal lossless converter is assumed, the instantaneous power on the dc side  $p_{dc}(t)$  should be equal to the instantaneous power on the ac side [4], thus,

$$p_{dc}(t) = p_{ac}(t). \quad (2.15)$$

This implies that both the dc component and the ripple power, are transferred from the ac side to the dc side. Inevitably, the ripple power now transferred onto the dc side generates second-order harmonic currents through the dc bus. The second-order harmonic currents, referred to as the *ripple current*, oscillates at a frequency  $f_r$ , similar to the frequency of the ripple power that generates it. The ripple current has the same amplitude as the *constant current* produced by the dc component.

Unfortunately, the generated ripple current flowing through the dc bus also flows through the battery. Ideally, the current flowing through the battery should be constant. Hence, the ripple current is of great concern in a single-phase dc-to-ac converter powered by a battery.

### The effect of ripple current on a battery

The temperature at which battery cells operate is a significant factor in determining its service life. The internal chemical reactions are driven either by the applied voltage or the operating temperature. Consider the case when the temperature is the driving parameter.

Heat is generated within the battery during the electrochemical exothermic reactions that occur within the cells, as well as when the current flows through the resistive elements of the cells (both the constant current and ripple current). This heat raises the temperature, increasing the rate of the chemical reactions and improving the performance of the battery. However, as the temperature continues to increase, so does the rate of unwanted chemical reactions. The unwanted chemical reactions, namely, the gassing and passivation of electrodes, lead to reduced battery life, affect the battery output capacity and reliability [22]. As a result, it leads to increased running costs due to increased maintenance or replacing the batteries earlier than planned or expected.

The major parameter that contributes to the increase in heat generation in a battery is the ripple current. Therefore, to avoid the excessive temperature rise when the chemical reactions occur, the recommended ripple current that flows through a battery should be limited to 10 % of the nominal battery current [5]. In the next section, the methods used for ripple energy compensation to reduce the ripple current passing through the battery are analyzed.

### 2.1.5 Analysis of capacitive ripple energy compensation

In an electric circuit, both capacitors and inductors can be used as energy storage components. Capacitors possess a higher energy density and have a lower weight than inductors. Assuming that the energy stored in an inductor is small compared to the energy stored in the capacitor, only capacitive energy storage will be analyzed in this thesis.

#### Passive capacitor for ripple energy compensation

The conventional method for ripple energy compensation makes use of a dc-link capacitor, which is denoted by  $C_{dc}$  in Figure 2.8. With an arbitrarily large dc-link capacitor, i.e., a capacitor with a high capacitance value, the battery current becomes relatively constant.

Firstly, before computing the minimum size of the dc-link capacitor required to reduce the ripple current to an acceptable value, it is necessary to analyze the ripple energy that needs to be stored or delivered by the dc-link capacitor.

For simplification, consider the instantaneous power transferred from the ac side to the dc side to have a unity power factor, i.e., the phase angle  $\phi = 0$ . Bear in mind that this simplification is used only to make the analysis of ripple energy apparent. The phase angle will be taken into account in the sections to follow since it is an extremely important parameter required for ripple energy compensation to be possible.

At unity power factor, the instantaneous power in (2.12), on the dc side, becomes

$$p_{dc}(t) = P_o - P_o \cos(2\omega_1 t), \quad (2.16)$$

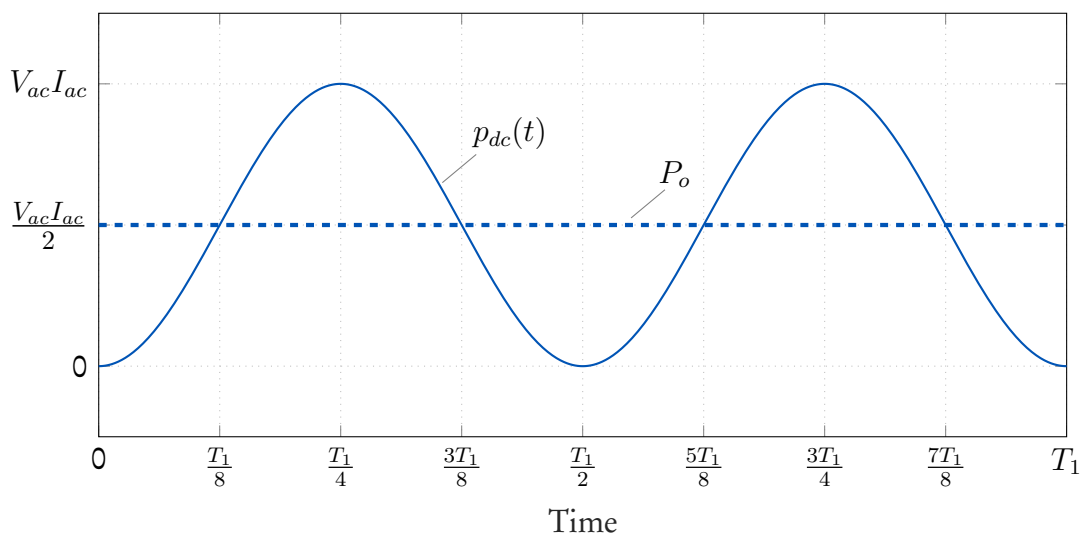
with

$$P_o = \frac{V_{ac} I_{ac}}{2}, \quad (2.17)$$

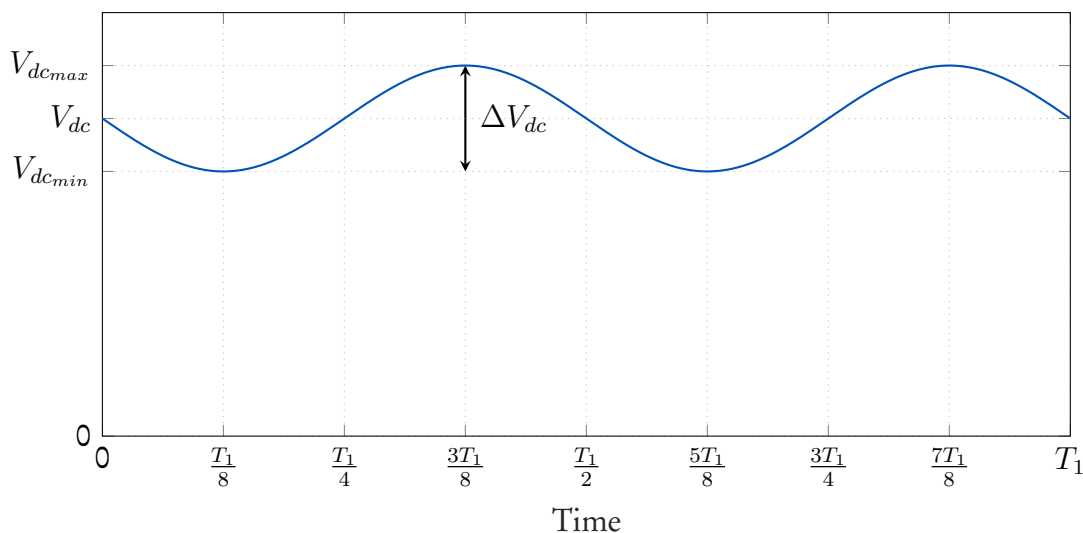
as the dc component.

In Figure 2.10, the key waveforms illustrating the pulsating power on the dc side and the way the dc-link capacitor stores and delivers energy in order to achieve ripple energy compensation are shown. Refer to both Figure 2.10a and Figure 2.10b for this analysis.

When  $p_{dc}(t) > P_o$  either from  $t = \frac{T_1}{8}$  to  $t = \frac{3T_1}{8}$  or from  $t = \frac{5T_1}{8}$  to  $t = \frac{7T_1}{8}$ , the voltage across the dc-link capacitor increases from  $V_{dc_{min}}$  to  $V_{dc_{max}}$ . This implies that the capacitor  $C_{dc}$  is charged during that time period. Conversely, when  $p_{dc}(t) < P_o$ , for example, from  $t = \frac{3T_1}{8}$  to  $t = \frac{5T_1}{8}$ , the voltage across the dc-link capacitor decreases from  $V_{dc_{max}}$  to  $V_{dc_{min}}$ , implying that the capacitor  $C_{dc}$  is discharging.



(a) The power  $p_{dc}(t)$  oscillating at a frequency  $f_r = 2f_1$ .



(b) The voltage  $v_{dc}(t)$  across the dc-link capacitor  $C_{dc}$ .

Figure 2.10: The key waveforms used for ripple energy analysis at unity power factor. The time axis is in segments of the fundamental period  $T_1 = \frac{1}{f_1}$ . Adopted from [23].

Thus, the ripple energy charging  $C_{dc}$  is deduced as

$$\begin{aligned} E_r &= \int_{\frac{T_1}{8}}^{\frac{3T_1}{8}} (p_{dc}(t) - P_o) dt \\ &= -P_o \int_{\frac{T_1}{8}}^{\frac{3T_1}{8}} \cos(2\omega_1 t) dt \\ &= \frac{P_o}{\omega_1}. \end{aligned} \quad (2.18)$$

We assume the dc supply voltage  $V_{dc}$  to be constant. Otherwise, the ripple energy in (2.18) would cause additional harmonics on the dc bus.

In general, the energy stored in a capacitor is given by  $E = \frac{1}{2}Cv^2$ . Thus, the energy stored in the dc-link capacitor  $C_{dc}$  is given by

$$E_{c_{dc}} = \frac{1}{2}C_{dc} (V_{dc_{max}}^2 - V_{dc_{min}}^2), \quad (2.19)$$

where

$$V_{dc_{min}} = V_{dc} - \frac{1}{2}\Delta V_{dc} \quad (2.20a)$$

$$V_{dc_{max}} = V_{dc} + \frac{1}{2}\Delta V_{dc}, \quad (2.20b)$$

denote the minimum value and maximum value of the voltage across the dc-link capacitor, respectively. With  $\Delta V_{dc}$  as the ripple component of the capacitor voltage, defined by

$$\Delta V_{dc} = V_{dc_{max}} - V_{dc_{min}}. \quad (2.21)$$

By using the law of conservation of energy, for ripple energy compensation to be possible, the energy stored in the dc-link capacitor should be equal to the ripple energy, that is,

$$E_{c_{dc}} = E_r. \quad (2.22)$$

From (2.22), the minimum capacitance required for ripple energy compensation is obtained as

$$C_{dc} = \frac{P_o}{\omega_1 V_{dc} \Delta V_{dc}}. \quad (2.23)$$

Since only a *small ripple component*  $\Delta V_{dc}$  is allowed across the dc-link capacitor, it can be seen that a relatively large dc-link capacitor will be required to compensate for the ripple energy.

The implication of employing the passive capacitor for ripple energy compensation can be demonstrated with an example. Consider a single-phase dc-to-ac converter with the rated parameters shown in Table 2.3.

Table 2.3: Parameters of a single-phase dc-to-ac converter.

Parameter	Symbol	SI value
Power rating	$P_o$	18 kW
Nominal dc voltage	$V_{dc}$	400 V
Ripple component	$\Delta V_{dc}$	8 V
Fundamental frequency	$f_1$	50 Hz

The maximum allowed ripple component  $\Delta V_{dc}$  is chosen as 2% of the nominal dc supply voltage  $V_{dc}$ . By using (2.23), the minimum capacitance required for the dc-link capacitor to reduce the ripple current,  $C_{dc} = 17.9$  mF. To meet this requirement, *electrolytic capacitors* can be used since they are available in large capacitance values. However, due to the short lifetime and large size of electrolytic capacitors, they contribute significantly to the cost and volume of the overall system. Therefore, it is important to refrain from using this type of capacitor to mitigate the ripple current.

### Alternative method of ripple energy compensation

The previous section described how the voltage across the dc-link capacitor is only allowed to have a small ripple component across it. As a consequence, a large capacitor is required to reduce the ripple current flowing through the battery to an acceptable value. In order to address the shortcoming arising from using a passive dc-link capacitor, an alternative method of ripple energy compensation is discussed in this section.

As already mentioned, the energy stored in a capacitor  $E = \frac{1}{2}Cv^2$ , with  $v$  as the voltage across the capacitor. If this voltage is allowed to fluctuate over a much wider range, more ripple energy will be compensated for.

The methods in [4, 7, 8] propose connecting an energy storage circuit on the dc side of a single-phase dc-to-ac converter. In Figure 2.11, a block diagram of the energy storage circuit connected between the points marked A and B is shown. It is highlighted in black.

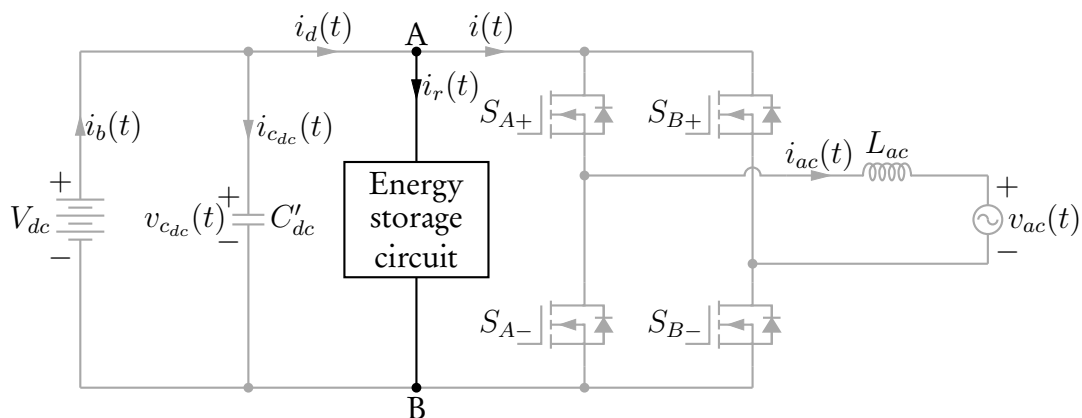


Figure 2.11: The concept of using an energy storage circuit for ripple energy compensation.

The energy storage circuit breaks the deadlock between reducing the ripple current and minimizing the size of the dc-link capacitor. In essence, it “replaces” the dc-link capacitor  $C_{dc}$ . As a substitute, a small dc-link capacitor  $C'_{dc}$  is used, as shown in Figure 2.11.

The energy storage circuit has an *active capacitor*, and the ripple current  $i_r(t)$  of  $i(t)$  is transferred to the active capacitor to keep the battery current  $i_b(t)$  relatively constant. In principle, an energy storage circuit is a dc-to-dc converter which can take the form of a bidirectional buck, boost or buck-boost converter. The circuit configurations of the different forms of dc-to-dc converters that can be connected between the point marked A and B in Figure 2.11, are shown in Figure 2.12.

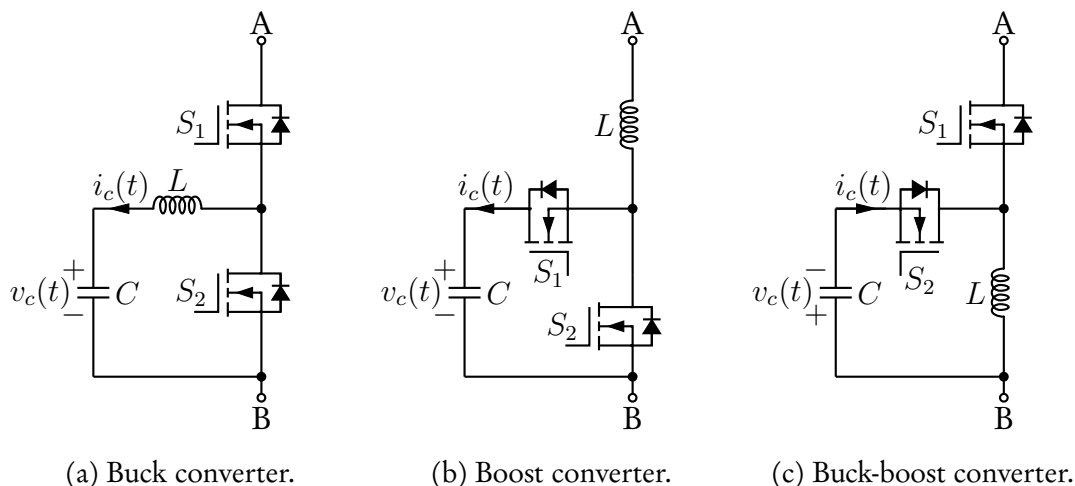


Figure 2.12: The circuit configurations of dc-to-dc converters.

Each of the dc-to-dc converters consists of two semiconductor switches denoted by  $S_1$  and  $S_2$ , an active capacitor  $C$ , and an inductor  $L$ . With the assumption that the inductor  $L$  stores a relatively small amount of energy, the instantaneous power that goes into the active capacitor is given by

$$p_c(t) = v_c(t)i_c(t), \quad (2.24)$$

where

$$i_c(t) = C \frac{dv_c(t)}{dt}, \quad (2.25)$$

is the current flowing through the active capacitor and  $v_c(t)$  is the voltage across it.

Suppose that the ripple power that goes to the active capacitor is given by [4],

$$p_r(t) = P_o \sin(2\omega_1 t - \phi). \quad (2.26)$$

If the power balance relationship is applied, then the instantaneous power that goes into the active capacitor is equal to the ripple power. Thus,

$$\begin{aligned} p_c(t) &= p_r(t) \\ v_c(t) C \frac{dv_c(t)}{dt} &= P_o \sin(2\omega_1 t - \phi). \end{aligned} \quad (2.27)$$

By solving (2.27), the active capacitor voltage can be derived as [4],

$$v_c(t) = \sqrt{\frac{P_o}{\omega_1 C} (k - \cos(2\omega_1 t - \phi))}, \quad (2.28)$$

where the coefficient

$$k \geq 1.$$

The coefficient value is chosen based on the type of dc-to-dc converter used as the energy storage circuit. For example, when the boost converter topology is chosen, then  $k$  is chosen such that the instantaneous voltage of the active capacitor is always higher than the dc bus voltage.

By definition, the energy stored in the active capacitor

$$E_c = \frac{1}{2} C v_c^2(t). \quad (2.29)$$

Recall that the ripple energy that needs to be compensated as given in (2.18) is

$$E_r = \frac{P_o}{\omega_1}.$$

By using the law of conservation of energy, the energy stored in the active capacitor should be equal to the ripple energy, that is,

$$E_c = E_r, \quad (2.30)$$

in order to reduce the ripple current flowing through the battery significantly. By solving (2.30), the minimum capacitance required when using an energy storage circuit for ripple energy compensation is obtained as

$$C = \frac{2P_o}{\omega_1 V_c^2}, \quad (2.31)$$

where  $V_c$  denotes the average voltage of the active capacitor. Since voltage  $v_c(t)$  across the active capacitor is allowed to swing over a wide range, it results in a high average voltage. Hence, by using (2.31) to calculate the minimum capacitance, a smaller capacitor will be required for ripple energy compensation.

In summary, the capacitance requirements are reduced by employing an active capacitor for ripple energy compensation instead of using the dc-link capacitor  $C_{dc}$ . If all the ripple current is diverted to the active capacitor, then the small dc-link capacitor  $C'_{dc}$  is required only to filter the high-frequency harmonic components caused by the switching of the single-phase dc-to-ac converter. Thus, instead of using electrolytic capacitors, *thin-film capacitors* can be used since they can handle higher ripple currents and have a much longer lifetime.

## 2.2 Active capacitor for ripple energy compensation

In the previous section, three different topologies of dc-to-dc converters used as energy storage circuits for ripple energy compensation were introduced. The choice of the topology used in this thesis and its working principle will be discussed in this section.

### 2.2.1 Analysis of dc-to-dc converter topologies

#### Buck converter and buck-boost converter

For the buck converter, the instantaneous voltage of the active capacitor should not exceed the dc supply voltage  $V_{dc}$ . In this regard, its average voltage  $V_c$  is always lower than  $V_{dc}$ . Thus, when using (2.31) to calculate the minimum capacitance required for ripple energy compensation, it can be seen that the reduction in the capacitance requirements will not be as significant. On the other hand, for the buck-boost converter, the instantaneous voltage of the active capacitor can be controlled to be either higher or lower than  $V_{dc}$ . If it is controlled to be higher, the buck-boost converter has an advantage over the buck converter in-terms of lower capacitance requirements.

Another factor considered when choosing the appropriate dc-to-dc converter topology is the ability to divert all the ripple current away from the battery to the energy storage circuit. Due to the semiconductor switch  $S_1$  directly connected to the dc bus for both the buck converter and buck-boost converter, not all the ripple current flows through the energy storage circuit when the switch  $S_1$  turns off. The simulation results presented in [24], only managed to reduce the ripple current flowing through the battery to 22.2% of the battery



current while utilizing the buck converter topology for ripple energy compensation. Thus, having direct control of the ripple current when entering the energy storage circuit is crucial, as will be proven at a later stage.

### Boost converter

The instantaneous voltage of the active capacitor for the boost converter must be higher than  $V_{dc}$ . Thus, the average voltage  $V_c$  is always higher than  $V_{dc}$ . As a result, the boost converter has the advantage of having the least capacitance requirements than the other two topologies. Furthermore, the boost converter topology has its inductor  $L$  directly connected to the dc bus, which allows direct control of the inductor current. For the reasons mentioned above, the boost converter topology is chosen as the energy storage circuit used in this thesis.

It is important to note that all of the three dc-to-dc converter configurations in Section 2.1.5 have a similar operating principle. However, the following section will be solely dedicated to the operation of an energy storage circuit based on the boost converter topology. For more information on the operation of energy storage circuits based on the buck and buck-boost converter topologies, the interested reader is referred to [4] and [7], respectively.

## 2.2.2 The operation principle of the dc-to-dc boost converter and control problem formulation

Figure 2.13 shows the grid-connected single-phase dc-to-ac converter with a dc-to-dc boost converter connected on the dc side.

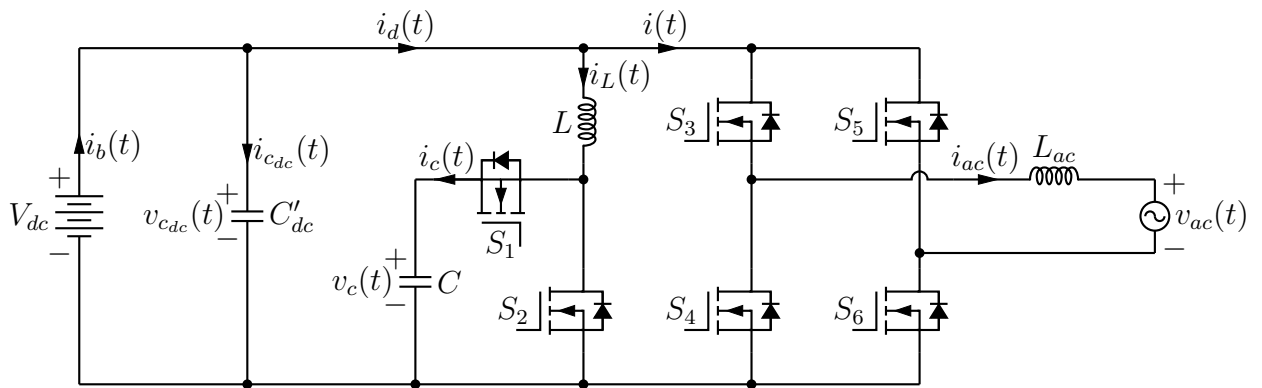


Figure 2.13: Grid-connected single-phase dc-to-ac converter with a boost converter connected on the dc side.

As mentioned in Section 2.1.5, the ripple current  $i_r(t)$  of  $i(t)$  needs to be diverted away from the battery and directed towards the active capacitor  $C$ . To achieve this, the inductor current  $i_L(t)$  flowing through the boost converter should accurately track the ripple current. The ripple current is given by

$$i_r(t) = -\frac{P_o}{V_{dc}} \cos(2\omega_1 t - \phi). \quad (2.32)$$

The underlying control objective of the boost converter is to divert  $i_r(t)$  away from the battery to keep the nominal battery current  $i_b(t)$  relatively constant. Also, the active capacitor

voltage  $v_c(t)$  should be controlled such that it fluctuates over a wide range while being charged by the ripple current and discharging through the dc bus. From this, a control problem is formulated whereby the switch positions of the boost converter,  $S_1$  and  $S_2$ , must be manipulated appropriately for  $i_L(t)$  to track the ripple current closely.

The two semiconductor switches,  $S_1$  and  $S_2$ , are switched in a complementary manner. Implying that two gating signals that are the inverse of one another control the switches. Thus, the converter is operated in continuous conduction mode (CCM). Different modes of operation of the boost converter can be distinguished, depending on the direction of flow of the ripple current. Recall that, for simplification, we assume that the drain-to-source voltage  $V_{DS}$  of the MOSFETs is greater than the on-state voltage  $V_{on}$  of the anti-parallel diode. Thus, current will only flow through the resistive region of the MOSFET instead of the anti-parallel diode.

Figure 2.14 depicts positive ripple current flow. The current path is highlighted in black. During positive current flow, the active capacitor is charged. Thus, the ripple energy is stored in the active capacitor.

Figure 2.15 depicts negative ripple current flow. The current path is highlighted in black. When the current flowing through the boost converter is negative, the active capacitor is discharged. The capacitor discharges by releasing the stored ripple energy through the dc bus.

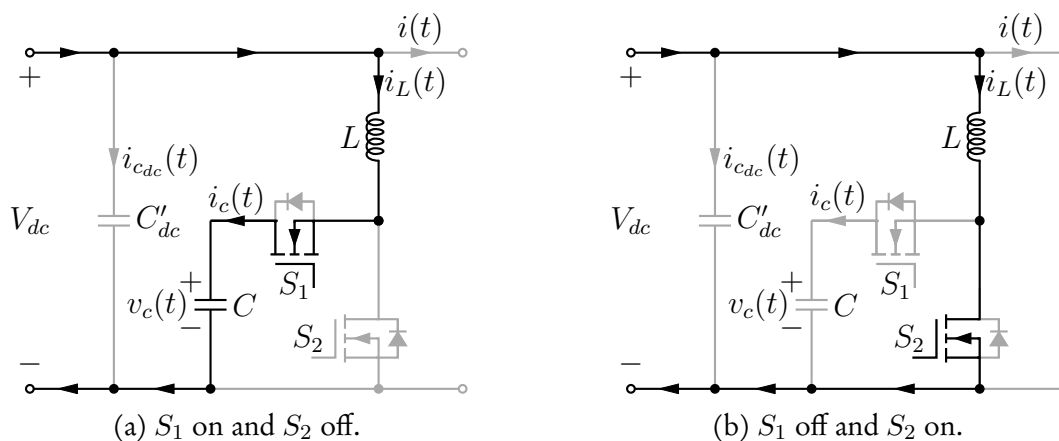


Figure 2.14: Positive current flow through the boost converter.

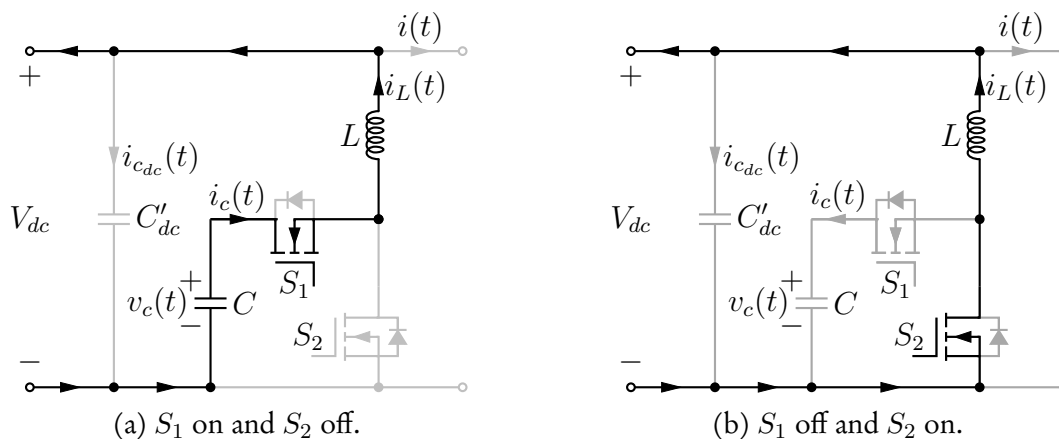


Figure 2.15: Negative current flow through the boost converter.

### 2.2.3 Proposed control schemes

As already stated, in order to divert the ripple current away from the battery to the active capacitor, the semiconductor switches must be manipulated in such a way that the current flowing through the energy storage circuit closely tracks the ripple current. In this regard, a suitable control strategy is needed. The existing literature has proposed different control strategies that have proven to be reasonably effective. The control strategies are based mainly on the classic linear controllers combined with nonlinear techniques, for instance, PWM.

In [8], a repetitive controller is proposed. It is used in cascade with a PI controller. The operation of an energy storage circuit relies on a properly regulated active capacitor voltage to allow a significant amount of ripple energy. Firstly, the average voltage across the active capacitor is obtained by passing it through a low pass filter, and it is regulated to a given reference value by using the PI controller. When the second-order harmonic component is extracted from the dc bus using a band-pass filter, it is added to the output of the PI controller. From this, the reference ripple current is obtained.

The repetitive controller, which consists of an internal model, is adopted to track the inductor current along with its reference, i.e., the generated ripple current, while rejecting any disturbances. The internal model of the controller comprises of a local positive feedback loop that has a delay term and is cascaded with a low-pass filter. The low-pass filter introduces a high gain at the frequency of the second-order harmonic component and improves the gain stability of the controller [25]. The gain is obtained via the  $H_\infty$  control design [25] or tuned through trial-and-error. The repetitive controller is able to eliminate periodic errors, since the reference ripple current is periodic in nature. The output of the repetitive controller is sent to a pulse-width modulator which in turn produces gating signals to the semiconductor switches of the energy storage circuit.

In [7], the same principle of first passing the active capacitor voltage through a low pass filter is followed. This is done to remove any voltage ripples possibly transferred from the dc bus to the active capacitor. To regulate the capacitor voltage, three different control methods all based on a PI controller are used. A dc-bias current required to charge the active capacitor is generated at the output of the PI controllers and used as the dc offset to the reference ripple current. The generated reference ripple current is then used to calculate the PWM duty cycle for the charging and discharging operation modes of the active capacitor. The output of the pulse-width modulator is used as the control input to the energy storage circuit.

Although the control strategies mentioned above have proven effective, there are several challenges associated with the design of the classical linear controllers. Usually, a PI-type controller is a common choice, evidently, from the control strategies mentioned earlier. One challenge in particular is the tuning of the parameters. The controller is tuned based on the linear state-space average model of the converter [26]. Furthermore, the obtained parameters used to achieve optimal performance only apply to a narrow operating range; when used outside of this range, the performance of the system significantly declines [26].

Another challenge is the ease of controller design. The aim is not only to improve the performance of the closed-loop system but also to have a systematic design approach to the control problem and implementation procedure. From this only, it can be deduced that the use of classical linear controllers presents challenges for both their theoretical and practical application.

To meet the ever-increasing demand for improved system performance and controllers that are easy to implement, other control strategies have been introduced. In the next part of the chapter, an alternative control strategy for ripple energy compensation while using an energy storage circuit will be discussed in detail.

---

## Part II

---

In this part of the chapter, a formal introduction to model predictive control is given. The fundamental principles and components on which it is based are presented. Moreover, a detailed explanation and the relevant examples of the optimization methods, namely, the branch-and-bound technique, and the move blocking strategy used to solve the optimization problem underlying direct long-horizon model predictive control, are given.

### 2.3 Model predictive control

#### 2.3.1 Introduction to MPC

In the 1970s, the process industry developed an advanced control strategy known as model predictive control [27]. Since its inception in the process industry, MPC has only recently emerged in the field of power electronics. This emergence is due to the advent of fast microprocessors and FPGAs with increased computational power.

MPC is a model-based control strategy, as the name insinuates. It predicts the future states and controlled variables of a sampled system, over a prediction horizon, by using the internal dynamic model of the system. Based on a cost function that captures the control objectives, an optimal control problem is formulated over the prediction horizon. By minimizing the cost function at every time-step, in real-time, subject to the evolution of the sampled system and constraints, the optimization problem is solved. The solution yields a control sequence that results in the optimal behaviour of the system. Out of the control sequence, only the first control action is applied to the system. At the next sampling instant, with the prediction horizon shifted, the state variables are updated based on new records of measurements, and the optimization procedure is repeated. This procedure is known as the *receding horizon policy*, and it provides feedback to the system.

#### Advantages of MPC

In a power electronic system, the most common building blocks are linear elements, namely, capacitors, inductors and resistors, accompanied by non-linear semiconductor switches. In this regard, for various combinations of the switch positions, linear functions of time that describe each of the combinations can be formulated. Thus, different dynamics of the system arise. As a consequence, it becomes intrinsically difficult to control the variables of such a system, for instance, controlling the voltage and current.

For example, dc-to-dc converters consist of a switched (continuous-time) linear system, also referred to as a *hybrid* system. Hybrid systems switch between different modes of operation, and a characteristic dynamic law governs each of the operation modes [26]. Usually, the hybrid systems are modelled by using averaging techniques as a way of concealing their switching behaviour [11]. Depending on the dynamics of the system or constraints imposed on the system, the resulting model is either linear or nonlinear. Nonlinearities can also be

caused by saturation effects of magnetic components, such as inductors. If classical linear controllers are considered for a nonlinear system, or a system that has constraints, the design effort increases.

MPC addresses these challenges. It is formulated in the time domain instead of the frequency domain, thereby incorporating any nonlinearities in the internal dynamic model of the system [11]. Also, MPC can systematically cope with constraints imposed on inputs, states, and outputs of the system, unlike PI controllers, which require anti-windup mechanisms to avoid the saturation of the integrator [11]. MPC imposes constraints on any of the variables without the need of additional control loops.

For multiple-input multiple-output (MIMO) systems, MPC requires only a single control loop. As opposed to the classical controllers, the MIMO system has to be broken down into multiple and ideally coupled single-input single-output (SISO) control loops. Furthermore, individual PI controllers must be designed for each SISO control loop. The design procedure of each of these control loops might appear to be easy and straightforward, however, in practice they tend to interact with each other in an adverse manner, thus, limiting the performance of the closed-loop system, particularly during transients [11].

### Indirect MPC versus direct MPC

The control strategies employed in most power electronic systems include a modulation stage. A continuous reference signal is modulated using a carrier-based pulse-width modulator or a space-vector modulator to generate gating signals for the semiconductor switches [11]. In Section 2.1.3, the concept of PWM is discussed in detail.

The use of PWM, in conjunction with MPC, formulates an indirect MPC strategy known as continuous-control set MPC (CCS-MPC) [11]. However, the modulation stage is nonlinear and it must be incorporated into the internal dynamic model of the system to account for its switching behaviour. As a result, a non-convex optimal control problem is formulated [28]. The solution to this problem prefers a global minimum, yet computing the global minimum in real-time is not practically feasible [28].

Upon removal of the modulator, a direct MPC strategy arises, also referred to as *finite-control set* MPC (FCS-MPC) [11]. According to FCS-MPC, the converter semiconductor switches are directly manipulated. Implying the direct application of the control action to the converter in the absence of a modulation stage. The internal dynamic model of the system is linear and the switching characteristics of the system are taken into account by the controller. The resulting optimization problem becomes an integer problem, i.e., a non-convex optimization problem. Fortunately, the optimization problem can be solved efficiently, even though it is non-convex, with a branch-and-bound technique and a move blocking strategy. In the following sections, only FCS-MPC will be considered, hereafter only referred to as MPC.

### 2.3.2 Internal dynamic model

The basic concept of MPC lies in using the internal dynamic model of a system that needs to be controlled in order to calculate the predicted output at future states for a given sequence of manipulated variables.

Consider the block diagram of a general control loop shown in Fig 2.16. It consists of an input vector  $\mathbf{u}(t) \in \mathbb{R}^{n_u}$  and an output vector  $\mathbf{y}(t) \in \mathbb{R}^{n_y}$ . By manipulating the switch positions of the system using  $\mathbf{u}(t)$ , and comparing the measured output  $\mathbf{y}(t)$  to its reference  $\mathbf{y}_{ref}(t)$  through a feedback loop, the output  $\mathbf{y}(t)$  is regulated along with its reference  $\mathbf{y}_{ref}(t)$ .

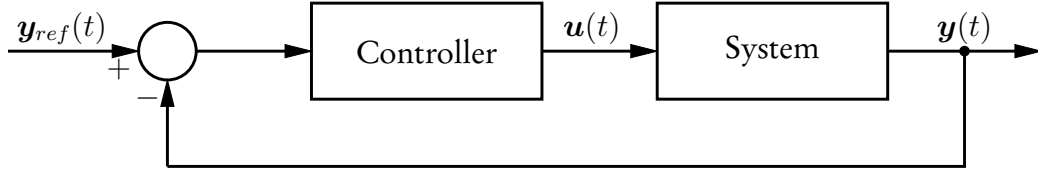


Figure 2.16: Block diagram of the basic control loop.

Let  $\mathbf{x}(t) \in \mathbb{R}^{n_x}$  denote the state vector of the system. Assuming that the system in Fig 2.16 can be described as a linear time-invariant system, the derived continuous-time state-space representation

$$\dot{\mathbf{x}}(t) = \mathbf{F}(\mathbf{u}(t))\mathbf{x}(t) + \mathbf{G}\mathbf{u}(t) \quad (2.33a)$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t), \quad (2.33b)$$

with  $\mathbf{F}(\mathbf{u}(t)) \in \mathbb{R}^{n_x \times n_x}$ ,  $\mathbf{G} \in \mathbb{R}^{n_x \times n_u}$ , and  $\mathbf{C} \in \mathbb{R}^{n_y \times n_x}$  as the state, input, and output matrix, respectively. The dimensions of the matrices are given by  $n_{subscript}$ , with subscripts  $x$ ,  $u$ , and  $y$  corresponding to the state, input, and output vectors, respectively.

In MPC, the manipulated variable can only change its value at discrete sampling instants, i.e., at time instants  $t = kT_s$ , where  $k \in \mathbb{N}$  and  $T_s$  is the sampling interval. The model representation in (2.33) can be discretized to obtain the discrete-time state-space representation

$$\mathbf{x}(k+1) = \mathbf{A}(\mathbf{u}(k))\mathbf{x}(k) + \mathbf{B}(\mathbf{u}(k))\mathbf{u}(k) \quad (2.34a)$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k), \quad (2.34b)$$

where

$$\mathbf{A}(\mathbf{u}(k)) = e^{\mathbf{F}(\mathbf{u}(k))T_s} \quad (2.35a)$$

$$\mathbf{B}(\mathbf{u}(k)) = \int_0^{T_s} e^{\mathbf{F}(\mathbf{u}(k))\tau} d\tau \mathbf{G}. \quad (2.35b)$$

If  $\mathbf{F}(\mathbf{u}(k))$  is non-zero, then  $\mathbf{B}(\mathbf{u}(k))$  in (2.35) can be expressed as

$$\mathbf{B}(\mathbf{u}(k)) = -\mathbf{F}(\mathbf{u}(k))^{-1}(\mathbf{I}_{n_x} - \mathbf{A}(\mathbf{u}(k)))\mathbf{G}. \quad (2.36)$$

Note that  $e$  refers to the matrix exponential and  $\mathbf{I}_{n_x}$  is an identity matrix with dimension  $n_x$ . This discretization method is referred to as exact discretization [11].

### 2.3.3 Constraints

Imposing constraints on variables of concern prevents the system from operating outside of its safe operating limits and physical limits during the optimization procedure. Constraints classified as either *hard* or *soft* constraints can be imposed on the input, state, and output variables,

$$\mathbf{u}(k) \in \mathcal{U} \subseteq \mathbb{R}^{n_u} \quad (2.37a)$$

$$\mathbf{x}(k) \in \mathcal{X} \subseteq \mathbb{R}^{n_x} \quad (2.37b)$$

$$\mathbf{y}(k) \in \mathcal{Y} \subseteq \mathbb{R}^{n_y}, \quad (2.37c)$$

which results in a nonlinear system [11]. Hard constraints represent the physical limitations of the system, and thus, cannot be violated. On the contrary, soft constraints are usually desirables that are imposed by the user, therefore, they can be slightly violated to avoid the control problem from becoming impracticable [11].

Consider a dc-to-dc boost converter with two controllable semiconductor switches that are switched complementary to each other. This implies that both switches cannot be on or off concurrently. This characteristic can be imposed as an input constraint

$$\mathcal{U} = \{0, 1\}. \quad (2.38)$$

Note that the constraint in (2.38) defines the only switch positions the converter can assume. Thus, it is classed as a hard constraint since it possesses physical limits.

### 2.3.4 Optimal control problem

The underlying control problem lies in designing a controller that achieves a given set of control objectives. The objectives are usually related to reference tracking and the switching effort. The switching effort is an indirect way to achieve a desired switching frequency.

In MPC, to formulate an optimal control problem, the control objectives as mentioned above are translated into a cost function. By solving the optimization problem, the aim is to find a sequence of manipulated variables that result in the optimal behaviour of the system.

#### Cost function

A general cost function is constructed as

$$J(\mathbf{x}(k), \mathbf{U}(k)) = \sum_{l=k}^{k+N_p-1} \Lambda(\mathbf{x}(l), \mathbf{u}(l)), \quad (2.39)$$

where  $\Lambda(\cdot, \cdot)$  represents the *weighting functions* over  $N_p$  time steps [11]. The  $N_p$  denotes a finite number of discrete-time steps over which the behaviour of the system is predicted. The time interval in which the prediction occurs is referred to as the *prediction horizon* with length  $N_p T_s$ . The arguments of the cost function are the current state vector  $\mathbf{x}(k)$  and the switching sequence

$$\mathbf{U}(k) = [\mathbf{u}^T(k) \quad \mathbf{u}^T(k+1) \quad \dots \quad \mathbf{u}^T(k+N_p-1)]^T, \quad (2.40)$$

introduced as a set of control commands at each time step over the prediction horizon.

The cost function in (2.39) is based on the *p-norm*, such as the 1-norm, 2-norm or  $\infty$ -norm [11]. Using the 1-norm or the  $\infty$ -norm, results in a linear cost function. For the 2-norm a quadratic cost function is constructed. The benefits of using a quadratic rather than a linear cost function are more appealing for predictive controllers with reference tracking, and thus from here on, only the quadratic cost function is considered. For a more detailed explanation of the choice of norms, the interested reader is referred to [11].

A quadratic cost function is given by

$$J(\mathbf{x}(k), \mathbf{U}(k)) = \sum_{l=k}^{k+N_p-1} \|\mathbf{y}_{ref}(l+1) - \mathbf{y}(l+1)\|_Q^2 + \lambda_u \|\Delta \mathbf{u}(l)\|_2^2. \quad (2.41)$$

The squared 2-norm vector  $\boldsymbol{\xi}$  (i.e., the first term in (2.41)) weighted with the penalty matrix  $\mathbf{Q} \in \mathbb{R}^{n_y \times n_y}$ ,

$$\|\boldsymbol{\xi}\|_Q^2 = \boldsymbol{\xi}^T \mathbf{Q} \boldsymbol{\xi}, \quad (2.42)$$

penalizes the deviation of the output vector  $\mathbf{y}(l)$  from its reference vector  $\mathbf{y}_{ref}(l)$ , that is, the predicted tracking error. The squared norm vector requires the penalty matrix  $\mathbf{Q}$  to be positive definite and symmetric, otherwise it does not induce a norm [11].

The second term in (2.41) is defined as

$$\Delta \mathbf{u}(l) = \mathbf{u}(l) - \mathbf{u}(l-1), \quad (2.43)$$

which refers to the switch positions. The weighting factor  $\lambda_u$  on the switching effort is a non-negative scalar, i.e.,  $\lambda_u \in \mathbb{R}$ . See Appendix A for general definitions of the norm relative to a vector and the positive definiteness of a matrix.

The diagonal entries of the penalty matrix  $\mathbf{Q}$  are used to prioritize the minimization of the tracking error of the output variables. In the case of controlling a dc-to-dc converter, priority can be given to either the inductor current or the capacitor voltage. For example, if the tracking accuracy of the inductor current is considered more important relative to that of the capacitor voltage, then a more substantial penalty is imposed on the inductor current tracking error and the capacitor voltage tracking error is penalized less.

The weighting factor  $\lambda_u$  sets the trade-off between the tracking error and the switching effort (a direct measure of the switching frequency or the switching losses [11]). By tuning  $\lambda_u$  accordingly, between  $0 \leq \lambda_u \leq \infty$ , the trade-off is adjusted. As  $\lambda_u \rightarrow 0$ , the switching transitions become less restricted; hence, the tracking error is prioritized more [12]. Conversely, as  $\lambda_u \rightarrow \infty$ , the switching transitions become more restricted; hence, the tracking error is prioritized less [12].

Therefore, the switching frequency  $f_{sw}$  of the system is adjusted by  $\lambda_u$ . It is important to note that the switching frequency for direct MPC is *variable*. Thus,  $f_{sw}$  refers to the average switching frequency of the system. Direct MPC inherits the challenge of choosing the appropriate values of the weighting factors since they are not always apparent.

### Optimization stage

By minimizing the cost function, subject to the discretized internal dynamic model of the system over the prediction horizon, and imposing constraints on the input variable, the solution to the optimization problem is obtained. The result of the optimization problem is the optimal sequence  $\mathbf{U}_{opt}(k)$  of the manipulated variables. The optimization problem is stated as follows

$$\mathbf{U}_{opt}(k) = \arg \min_{\mathbf{U}(k)} J(\mathbf{x}(k), \mathbf{U}(k)) \quad (2.44a)$$

$$\text{subject to } \mathbf{x}(l+1) = \mathbf{A}(\mathbf{u}(l))\mathbf{x}(l) + \mathbf{B}(\mathbf{u}(l))\mathbf{u}(l) \quad (2.44b)$$

$$\mathbf{y}(l+1) = \mathbf{C}\mathbf{x}(l+1) \quad (2.44c)$$

$$\mathbf{u}(l) \in \mathcal{U}, \quad (2.44d)$$

with  $l = k, \dots, k + N_p - 1$ . The optimal solution obtained at time step  $k$  is

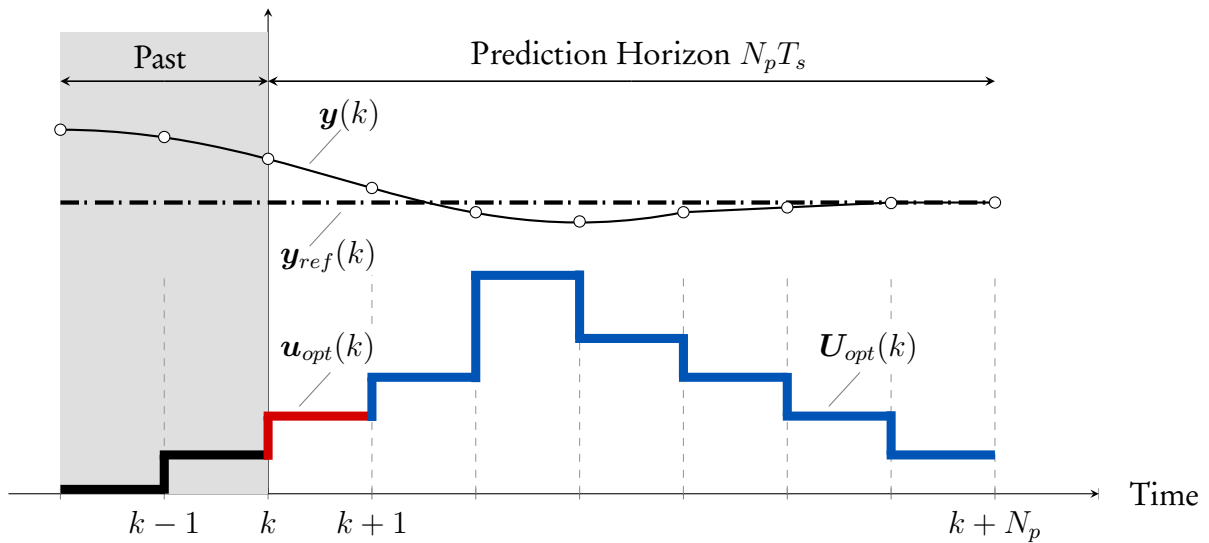
$$\mathbf{U}_{opt}(k) = [\mathbf{u}_{opt}^T(k) \quad \mathbf{u}_{opt}^T(k+1) \quad \dots \quad \mathbf{u}_{opt}^T(k+N_p-1)]^T, \quad (2.45)$$

which results in the optimal behaviour of the system. It should be mentioned that, the optimal solution in (2.45) is an *open-loop* solution.

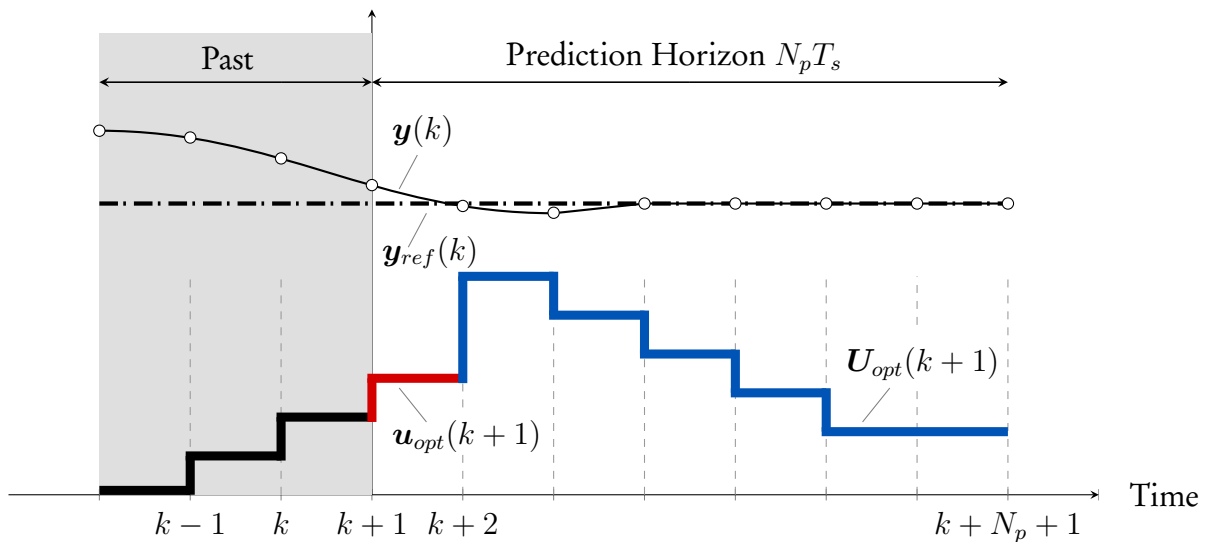


### 2.3.5 Receding horizon policy

The optimal switching sequence  $\mathbf{U}_{opt}(k)$  in (2.45) is a result of the open-loop optimization problem in (2.44), as mentioned in Section 2.3.4. In any case, the system requires a *closed-loop* control scheme to cope with uncertainties and disturbances that may occur, making the system more robust. Thus, the so-called receding horizon policy is applied to provide feedback [11]. It works as follows; out of the optimal switching sequence  $\mathbf{U}_{opt}(k)$ , only the first element,  $\mathbf{u}_{opt}(k)$ , is applied to the system at the sampling instant  $k$ . At the next sampling instant  $k + 1$ , the procedure is repeated. That is, by setting  $k$  to  $k + 1$  and obtaining new measurements, the optimization problem in (2.44) is solved over the shifted prediction horizon. The principle of the receding horizon policy is illustrated in Figure 2.17.



(a) Prediction horizon at the sampling instant  $k$ . The time  $t = kT_s$ .



(b) Prediction horizon at the sampling instant  $k + 1$ . The time  $t = (k + 1)T_s$ .

Figure 2.17: Illustrative example of the principle of the receding horizon policy. The optimal switching sequence  $\mathbf{U}_{opt}(k)$  is computed over a prediction horizon of  $N_p$  and only the first element  $\mathbf{u}_{opt}(k)$  is applied. The predicted output  $\mathbf{y}(k)$  tracks the given reference  $\mathbf{y}_{ref}(k)$ . At the next sampling instant, the procedure is repeated and the sequence  $\mathbf{U}_{opt}(k + 1)$  is obtained.

### 2.3.6 MPC based on exhaustive search

In many instances, the underlying optimization problem (2.44) is solved via the enumeration of all the possible switching sequences that can be assumed by  $\mathbf{U}(k)$ . All the possible switching sequences are given by  $\mathbf{U}(k) \in \mathcal{U}^{N_p}$ , also referred to as *candidate solutions*. For each of these switching sequences, the cost function is computed. The switching sequence that results in the lowest cost is the optimal solution, and its first element is applied to the system. This concept is well-known as *exhaustive search*. Algorithm 1 shows its implementation.

---

#### Algorithm 1 Exhaustive search

---

```

1: function  $\mathbf{u}_{opt}(k) = \text{MPC}(\mathbf{x}(k), \mathbf{u}(k-1))$ 
2:    $J_{opt}(k) = \infty, \mathbf{U}_{opt}(k) = \emptyset, \mathbf{u}_{opt}(k) = \emptyset$ 
3:   for all  $\mathbf{U}(k)$  over  $N_p$  do
4:      $J = 0$ 
5:     for  $l = k$  to  $k + N_p - 1$  do
6:        $\mathbf{x}(l+1) = \mathbf{A}(\mathbf{u}(l))\mathbf{x}(l) + \mathbf{B}(\mathbf{u}(l))\mathbf{u}(l)$ 
7:        $J = J + \|\mathbf{y}_{ref}(l+1) - \mathbf{y}(l+1)\|_Q^2 + \lambda_u \|\Delta\mathbf{u}(l)\|_2^2$ 
8:     end for
9:     if  $J < J_{opt}(k)$  then
10:       $J_{opt}(k) = J, \mathbf{U}_{opt}(k) = \mathbf{U}(k), \mathbf{u}_{opt}(k) = \mathbf{U}_{opt}(1)$ 
11:    end if
12:  end for
13: end function

```

---

Below is the procedure followed by a model predictive controller based on exhaustive search:

1. Obtain the switch position  $\mathbf{u}(k-1)$  and measurements for  $\mathbf{x}(k)$ .
2. Execute Algorithm 1.
3. Apply  $\mathbf{u}_{opt}(k)$  to the system.
4. Set  $k = k + 1$ , by shifting the prediction horizon with one sampling instant.
5. Return to step 1.

### 2.3.7 Computational burden

The underlying optimization problem based on direct MPC with reference tracking forms its basis on decision variables that are constrained to be integer variables. The integer variables, due to their nature (i.e., hard constraints) result in a non-convex optimization problem which is difficult to solve. By adding more integer variables, that is, by extending the prediction horizon, the number of possible solutions increases exponentially.

As previously discussed in Section 2.3.6, the optimization problem can be solved using the exhaustive search method. However, exhaustively solving the optimization problem becomes computationally intractable upon increasing the prediction horizon. Hence, this method is computationally feasible only for short prediction horizons of  $N_p = 1$  or  $N_p = 2$ .

Fortunately, the non-convex optimization problem can be solved efficiently with a long prediction horizon by using a branch-and-bound technique [12]. In addition to that, the computations required can be further reduced, and at the same time, maintain a sufficiently long prediction horizon by employing a move blocking strategy [14]. The two optimization methods and relevant examples are discussed in the following two sections.

## 2.4 Branch-and-bound technique

### 2.4.1 Introduction to the branch-and-bound technique

The branch-and-bound technique seeks to reduce the number of candidate solutions investigated, instead of enumerating all of the candidate solutions as done with the exhaustive search method. The technique significantly reduces the computational burden of solving the optimization problem, and thus enabling the use of long prediction horizons in power electronic applications.

The branch-and-bound technique is performed through a depth-first traversal of a search tree with depth  $n = N_p$ . Figure 2.18 shows an example of a search tree. The branches of the search tree represent the admissible switch positions  $\mathcal{U} = \{0, 1\}$ , which are the elements of the switching sequence  $\mathbf{U}(k)$ . The admissible switching positions are considered at every level  $j$  of the search tree. Firstly, an initial upper bound cost is chosen. The details of how to obtain a good initial upper bound cost will follow

During the traversal of the search tree, the branches are explored. While exploring a particular branch, a node is visited, and the cumulative cost at that node is calculated. If the cumulative cost at that node exceeds the current upper bound, the branch is pruned. Implying that the uninvestigated candidate solutions of the current branch are suboptimal, and thus, no further exploration of the branch is required. After pruning the branch, the candidate solutions are reduced earlier in the search process, and the next branch is explored. When the bottom node, referred to as a *leaf node*, is reached while exploring a branch, then a full switching sequence has been assembled. If the cumulative cost at the leaf node does not exceed the current upper bound, the upper bound is updated, and the incumbent optimal solution is updated if it is still within the upper bound. It is important to note that a certificate of optimality can only be issued once the entire search tree has been explored.

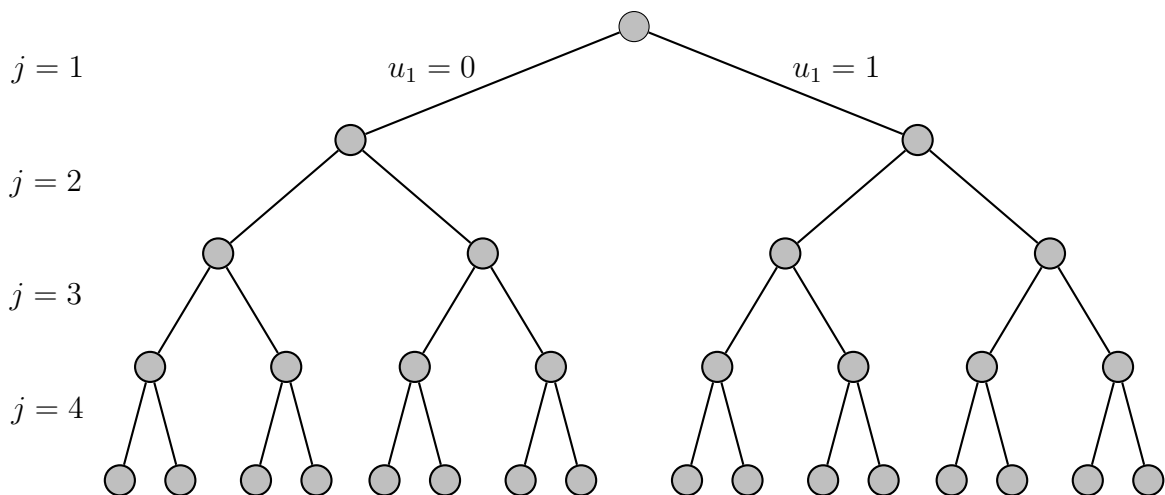


Figure 2.18: Example of a search tree with depth  $n = N_p$ . The entries of the switching sequence are represented by  $j$  which denotes the level in the search tree. The switch positions  $\mathcal{U} = \{0, 1\}$  are represented by the branches of the search tree.

### 2.4.2 The initial upper bound guess

To implement the branch-and-bound technique, and solve the optimization problem in a time-efficient manner, requires a good initial upper bound cost  $J_{ini}$  to warm-start the optimization

process. A method known as the *educated guess* can be used to determine a good initial upper bound cost [12]. It works in this way; the entries of  $\mathbf{U}_{opt}(k-1)$  are shifted forward by one sampling instant and repeating the last switch position. The guessed switching sequence

$$\mathbf{U}_{ini}(k) = \begin{bmatrix} \mathbf{0} & \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \dots & \dots & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \dots & \dots & \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{U}_{opt}(k-1),$$

with  $\mathbf{I}$  as the identity matrix, is obtained and used to calculate the initial upper bound cost  $J_{ini}$ . The educated guess forms its basis on the assumption that the switching sequence obtained at the next sampling instant matches the ones that were discarded by following the receding horizon policy. Therefore,  $\mathbf{U}_{ini}(k)$  is considered a feasible solution of the optimization problem (2.44).

### 2.4.3 Illustrative example of the branch-and-bound technique

Consider a dc-to-dc boost converter with a prediction horizon of  $N_p = 4$ . An example of a search algorithm in progress is shown Figure 2.19. The search algorithm demonstrates how the branch-and-technique traverses the search tree during the optimization procedure in order to determine the optimal switching sequence  $\mathbf{U}_{opt}(k)$  [12].

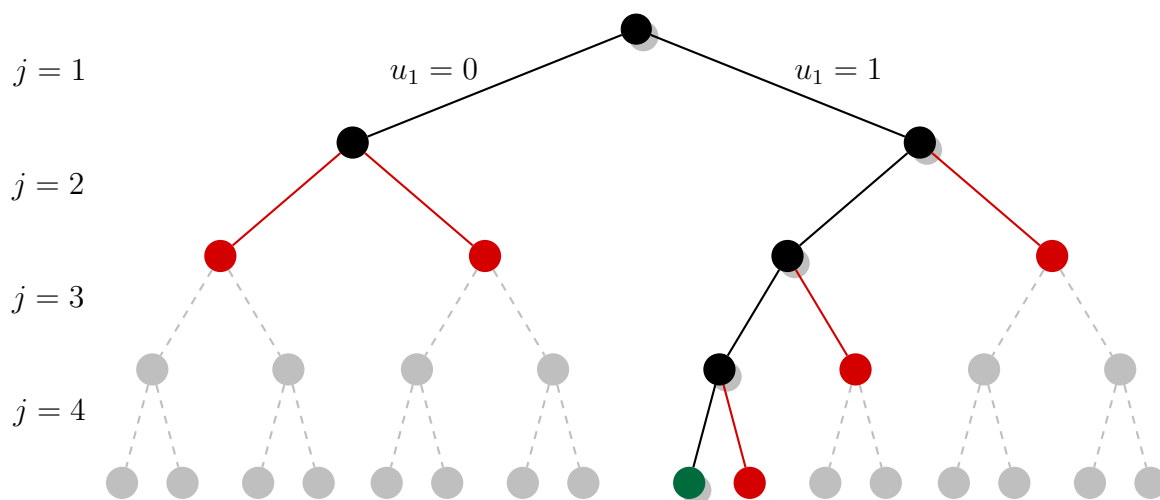


Figure 2.19: Example of a search tree with depth  $n = 4$ . The branches that have not been explored in the search process are denoted by the gray dashed lines. The red branches and nodes indicate the branches that were explored, however, the cumulative cost at the red node exceeded the upper bound. The branches that were traversed and evaluated are represented in black, and the green node represents the optimal solution. Adopted from [29].

The search algorithm traverses the tree from the left-to-right branch at any node in the search tree, that is, from switch position (0) to switch position (1), except for when it reaches the leaf nodes. Starting from the top of the search tree, also known as the *root node*, the initial upper bound cost is set to  $J_{ini}$ . The first element of  $\mathbf{U}(k)$  is  $u_1 = 0$  (i.e, a partial candidate solution) at the first level  $j = 1$  of the search tree. By using  $u_1 = 0$  and calculating the candidate

cost, it turned out that the candidate cost is less than the upper bound, and thus, the current branch should be explored further. Now moving to the second level of the search tree with  $j = 2$ , the second element is  $u_2 = 0$  and a partial switching sequence  $\mathbf{U}(k) = [0 \ 0]^T$  has been assembled. After calculating the candidate cost, it is found that the cumulative cost also exceeds upper bound; therefore, the branch is pruned, implying that there is no need to explore the branch any further. Thereafter, the next branch is considered with  $u_2 = 1$  and a partial switching sequence  $\mathbf{U}(k) = [0 \ 1]^T$ . Again the candidate cost is calculated and the cumulative cost compared to the upper bound. It turned out that the cumulative cost exceeds the upper bound. Therefore, the branch is pruned. Through backtracking, the second branch of the root node  $u_1 = 1$  at the first level  $j = 1$  is explored. The candidate cost is calculated, and it is found that it is less than the upper bound. The second element is  $u_2 = 0$ , with a partial switching sequence  $\mathbf{U}(k) = [1 \ 0]^T$ . The cumulative cost is calculated, and it is less than the upper bound. Branching further down the search tree, at the next level  $j = 3$  with  $u_3 = 0$ , the switching sequence is  $\mathbf{U}(k) = [1 \ 0 \ 0]^T$ . After calculating the cumulative cost, it turned out to be less than the upper bound. Therefore, the branch is explored further down to the element of the switching sequence  $u_4 = 0$ . Finally a full switching sequence  $\mathbf{U}(k) = [1 \ 0 \ 0 \ 0]^T$  is assembled. The cumulative cost is calculated, and it turned out that it is less than the upper bound, therefore, the upper bound is updated. The incumbent optimal solution is updated. The next branch with  $u_4 = 1$  is evaluated and it turns out that the cumulative cost exceeds the current upper bound. Through backtracking, the branch with  $u_3 = 1$  is investigated and it turned out that the cumulative cost exceeds the current upper bound, and the branch is pruned. The search algorithm backtracks to evaluate the branch on level  $j = 2$  with  $u_2 = 1$ . After calculating the cumulative cost, and it is found that the cumulative cost exceeds the current upper bound, and the remaining branch is pruned. A certificate of optimality is issued to the incumbent optimal solution  $\mathbf{U}_{opt}(k)$ .

In the practical implementation of the branch-and-bound technique on an FPGA, after the search algorithm has traversed the entire search tree and none of the options are viable in terms of cumulative cost, the best solution found during the search process will be used as the optimal solution. This will be further discussed in Chapter 5.

In [15], an algorithm that implements the branch-and-bound technique recursively is proposed. Algorithm 2 shows the implementation.

---

**Algorithm 2** Branch-and-bound technique
 

---

```

1: function  $\mathbf{u}_{opt}(k) = \text{MPC}(\mathbf{U}, \mathbf{x}(k), \mathbf{u}(l-1), J, J_{opt}, l)$ 
2:    $J_{opt}(k) = \infty, \mathbf{U} = \emptyset, \mathbf{U}_{opt}(k) = \emptyset, \mathbf{u}_{opt}(k) = \emptyset, k = l$ 
3:   for each  $\mathbf{u}(l) \in \mathcal{U}$  do
4:      $\mathbf{x}(l+1) = \mathbf{A}(\mathbf{u}(l))\mathbf{x}(l) + \mathbf{B}(\mathbf{u}(l))\mathbf{u}(l)$ 
5:      $J = J + \|\mathbf{y}_{ref}(l+1) - \mathbf{y}(l+1)\|_Q^2 + \lambda_u \|\Delta\mathbf{u}(l)\|_2^2$ 
6:     if  $J < J_{opt}(k)$  then
7:       if  $l < k + N_p - 1$  then
8:          $\text{MPC}(\mathbf{U}, \mathbf{x}(l+1), \mathbf{u}(l), J, J_{opt}, l+1)$ 
9:       else
10:         $J_{opt} = J, \mathbf{U}_{opt}(k) = \mathbf{U}(k), \mathbf{u}_{opt}(k) = \mathbf{u}_{opt}(1)$ 
11:      end if
12:    end if
13:  end for
14: end function

```

---

## 2.5 Move blocking strategy

### 2.5.1 Introduction to the move blocking strategy

Various power electronic applications require a long prediction horizon  $N_p T_s$  to successfully tackle a control problem. For example, a dc-to-dc boost converter requires a sufficiently long prediction horizon to account for the non-minimum phase behaviour it exhibits [26]. However, as mentioned earlier in Section 2.3.7, merely increasing the number of prediction steps  $N_p$  results in an exponential increase in the number of possible solutions, and consequently, an increase in the number of calculations required.

On the other hand, the sampling interval  $T_s$  could be increased while keeping the number of prediction steps few. Nevertheless, this is also not a viable solution. Since the switching occurs only at the sampling instants, by increasing the sampling interval, the resolution of the possible sampling instants becomes unacceptable.

To overcome the plausible contradiction of achieving a long prediction horizon either with an increase in the number of time-steps or a longer sampling interval, the so-called move blocking strategy can be employed [13, 14, 15].

### 2.5.2 Implementation of the move blocking strategy

The concept of the move blocking strategy involves splitting of the prediction horizon into two parts. The first part of the prediction horizon has the number of steps denoted by  $N_1$ , and for the second part, the number of steps are denoted by  $N_2$ , with  $N_1, N_2 \in \mathbb{N}^+$ , where  $\mathbb{N}^+$  refers to positive integers. For the entire prediction horizon, the total number of prediction steps  $N_p = N_1 + N_2$ .

The first part of the horizon with  $N_1$  steps is sampled finely with a sampling interval  $T_s$ , and the second part of the horizon with  $N_2$  steps is sampled more coarsely with a multiple of  $T_s$ , that is, with  $T'_s = n_s T_s$ , where  $n_s \in \mathbb{N}^+$ . As a result, the total length of the prediction horizon is a sum of the prediction lengths of the first and second part, which is given by  $N_1 T_s + N_2 T'_s = (N_1 + n_s N_2) T_s$ .

Note that high-resolution sampling of the system is required, particularly for the first  $N_1$  steps, mainly because the state transitions can only occur at sampling instants. Also, by sampling the steps  $N_2$  that are farther in the future more coarsely, obtains a long prediction interval. Thus, the desired goals are achieved, namely:

- Achieving a sufficiently long prediction horizon without a subsequent increase in the computational burden.
- Achieving a sufficiently long prediction horizon without decreasing the “effective” timing resolution required for accurate switching transitions.

### 2.5.3 Illustrative example of the move blocking strategy

To get a clearer understanding of the move blocking scheme, consider a system that can assume the switch positions  $\mathcal{U} = \{0, 1\}$  as the input vector. For a prediction horizon of  $N_p = 12$ , a switching sequence  $\mathbf{U}(k) = [1 \ 0 \ 1 \ 0 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 1 \ 1]^T$  shown in Figure 2.20a can be assumed. If a single sampling interval  $T_s$  is applied for the entire prediction horizon, the length of prediction horizon is  $12T_s$ , and the resulting output trajectory would be as shown in Figure 2.20b.

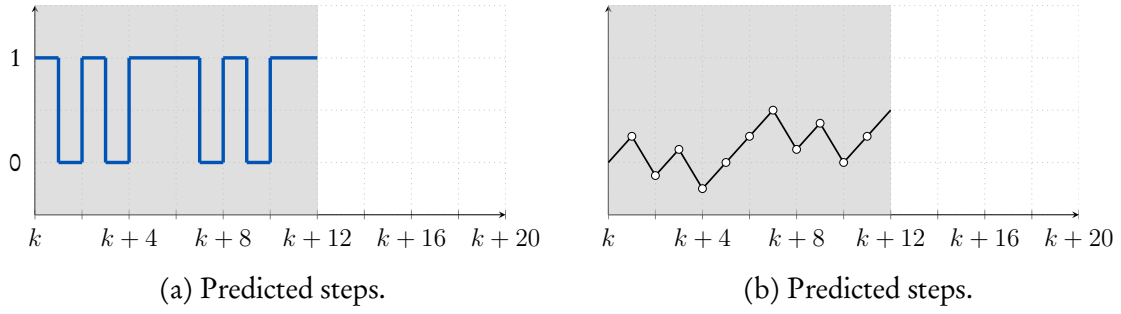


Figure 2.20: Without move blocking. Replicated from [13].

By implementing the move blocking scheme with  $N_1 = 8$ ,  $N_2 = 4$  and  $n_s = 3$ . The resulting switching sequence is shown in Figure 2.21a, and the output trajectory would be as shown in Figure 2.21b. Even though the prediction horizon still has the same number of time-steps  $N_p = 12$ , that is  $N_1 + N_2$ , the total prediction length of the prediction horizon is now  $20T_s$ .

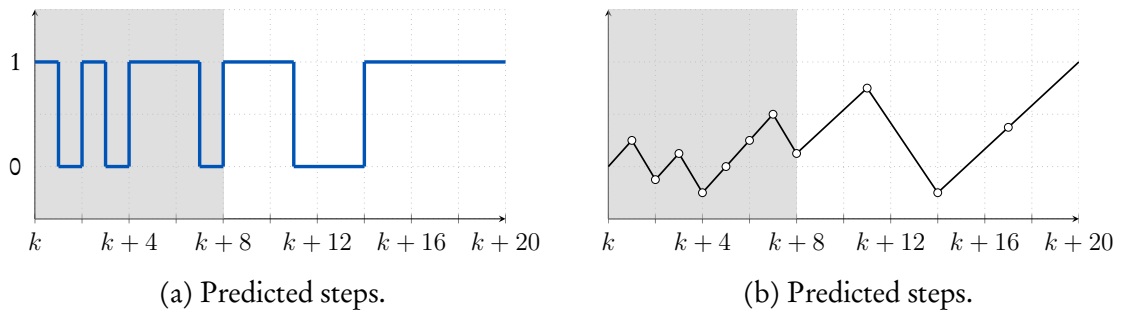


Figure 2.21: With move blocking. Replicated from [13].

In [13], an algorithm that implements the move blocking strategy based on the exhaustive search method is proposed. The algorithm is shown in Algorithm 3.

---

**Algorithm 3** Move blocking strategy
 

---

```

1: function  $\mathbf{u}_{opt}(k) = \text{MPC}(\mathbf{x}(k), \mathbf{u}(k-1))$ 
2:    $J_{opt}(k) = \infty$ ,  $\mathbf{U}_{opt}(k) = \emptyset$ ,  $\mathbf{u}_{opt}(k) = \emptyset$ 
3:   for all  $\mathbf{U}(k)$  over  $N_p$  do
4:      $J = 0$ 
5:     for  $l = k$  to  $k + N_p - 1$  do
6:       if  $l < k + N_1$  then
7:          $\mathbf{x}(l+1) = f_1(\mathbf{x}(l), \mathbf{u}(l))$ 
8:       else
9:          $\mathbf{x}(l+1) = f_2(\mathbf{x}(l), \mathbf{u}(l))$ 
10:      end if
11:       $J = J + \|\mathbf{y}_{ref}(l+1) - \mathbf{y}(l+1)\|_Q^2 + \lambda_u \|\Delta \mathbf{u}(l)\|_2^2$ 
12:    end for
13:    if  $J < J_{opt}(k)$  then
14:       $J_{opt} = J$ ,  $\mathbf{U}_{opt}(k) = \mathbf{U}(k)$ ,  $\mathbf{u}_{opt}(k) = \mathbf{U}_{opt}(1)$ 
15:    end if
16:  end for
17: end function

```

---

## Chapter 3

# Model predictive control of a dc-to-dc boost converter

### 3.1 Introduction

This chapter discusses the modelling of a dc-to-dc boost converter used to implement long-horizon direct MPC. The optimal control formulation for the boost converter is presented, with a detailed explanation of the difficulty encountered with controlling the active capacitor voltage. The benefits of using the move blocking strategy to alleviate the challenge faced are explained. A non-recursive algorithm that incorporates both the branch-and-bound technique along with the move blocking strategy is presented. The two optimization methods are used to reduce the computational burden of long prediction horizons for the practical implementation of the controller on an FPGA. Several symbols and concepts introduced in Chapter 2 are repeated in this chapter for convenience. The chapter concludes with a summary.

### 3.2 Modelling and control law formulation

To recap the study presented in this thesis: a dc-to-dc boost converter was introduced to divert the ripple current generated in a battery-powered single-phase dc-to-ac converter away from the battery to an active capacitor. This method of ripple energy compensation introduces a control problem. The proposed control strategy, referred to as long-horizon direct MPC, introduced in Section 2.3, is used to control the boost converter in order to solve the underlying control problem. Figure 3.1 shows the topology of the boost converter, which will be used as the system.

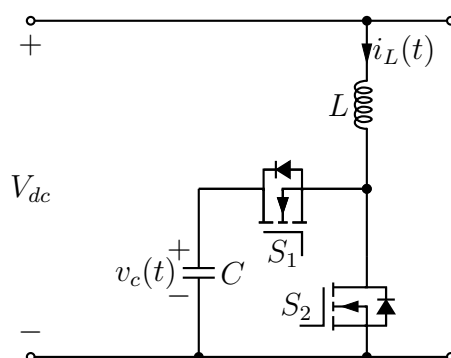


Figure 3.1: Topology of the dc-to-dc boost converter.



The dynamic evolution of the system is described by the continuous-time state-space representation in-terms of the switch position  $u$  as

$$\dot{\mathbf{x}}(t) = \begin{cases} \mathbf{F}_0 \mathbf{x}(t) + \mathbf{G}u(t), & S_1 = 0 \quad \& \quad S_2 = 1, \quad \text{Mode 1} \\ \mathbf{F}_1 \mathbf{x}(t) + \mathbf{G}u(t), & S_1 = 1 \quad \& \quad S_2 = 0, \quad \text{Mode 2} \end{cases} \quad (3.1)$$

where

$$\mathbf{x}(t) = [i_L(t) \quad v_c(t)]^T, \quad (3.2)$$

is the state vector which includes the inductor current and the active capacitor voltage. The state matrices are given by

$$\mathbf{F}_0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{F}_1 = \begin{bmatrix} 0 & -\frac{1}{L} \\ \frac{1}{C} & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{G} = \begin{bmatrix} \frac{V_{dc}}{L} \\ 0 \end{bmatrix}. \quad (3.3)$$

The switch positions of the individual switches are restricted to  $u \in \mathcal{U}$ , where  $\mathcal{U} = \{0, 1\}$ . Hence, the switching transitions operate as follows: when  $u = 1$ , switch  $S_1$  is *on* and  $S_2$  is *off*; and when  $u = 0$ , switch  $S_1$  is *off* and  $S_2$  is *on*. In Figure 3.2, a graphical summary of the continuous-time state-space representation of the boost converter is shown as an automation.

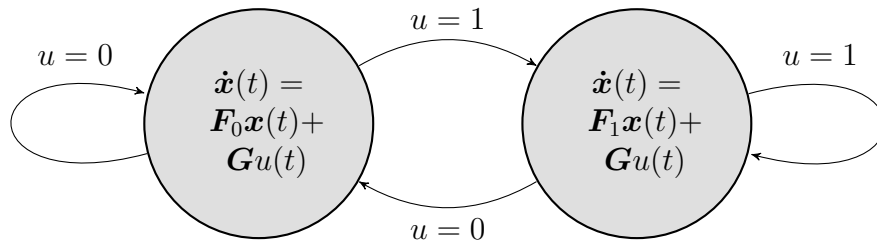


Figure 3.2: The continuous-time state-space representation of a boost converter presented as a continuous-time automation.

Using exact discretization [11], the continuous-time state-space representation of the system in (3.1) is discretized. The resulting discrete-time state-space representation of the system is given by

$$\mathbf{x}(k+1) = \begin{cases} \mathbf{A}_0 \mathbf{x}(k) + \mathbf{B}_0 u(k), & S_1 = 0 \quad \& \quad S_2 = 1, \quad \text{Mode 1} \\ \mathbf{A}_1 \mathbf{x}(k) + \mathbf{B}_1 u(k), & S_1 = 1 \quad \& \quad S_2 = 0, \quad \text{Mode 2} \end{cases} \quad (3.4)$$

where

$$\mathbf{A}_0 = \mathbf{I}_2 \quad (3.5a)$$

$$\mathbf{A}_1 = e^{\mathbf{F}_1 T_s} \quad (3.5b)$$

$$\mathbf{B}_0 = T_s \mathbf{G} \quad (3.5c)$$

$$\mathbf{B}_1 = -\mathbf{F}_1^{-1} (\mathbf{I}_2 - \mathbf{A}_1) \mathbf{G}, \quad (3.5d)$$

with  $T_s$  being the sampling period and  $e$  as the exponential matrix.  $\mathbf{I}_2$  denotes the  $2 \times 2$  identity matrix. The two different modes of operation of the boost converter are illustrated in Figure 3.3 as an automation for the discrete-time state-space representation.

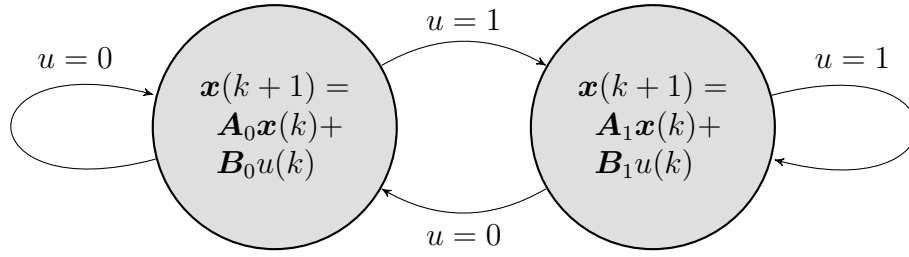


Figure 3.3: The discrete-time state-space representation of a boost converter presented as a discrete-time automation.

The control objective for the boost converter is two-fold. First, the inductor current  $i_L(t)$  should accurately track the ripple current  $i_r(t)$  in (2.32). It is important to note that, minimizing the current tracking error is the main control objective since all the ripple current needs to be diverted away from the battery. Second, the active capacitor voltage  $v_c(t)$  must be allowed to fluctuate over a wide range, as discussed in Section 2.1.5. Thus, the active capacitor voltage should be regulated along with the time-varying reference signal given in (2.28) to allow a significant amount of ripple energy through the charging and discharging of the capacitor.

By applying reference tracking to the controlled variables and penalizing the switching transitions, the formulated cost function over the prediction horizon of  $N_p$  is defined as

$$J = \sum_{l=k}^{k+N_p-1} \|\mathbf{x}_{ref}(l+1) - \mathbf{x}(l+1)\|_{\mathbf{Q}}^2 + \lambda_u \|\Delta u(l)\|_2^2, \quad (3.6)$$

where  $\mathbf{x}_{ref}$  is the vector encompassing the reference signals of the state vector, that is, the controlled variables; hence,

$$\mathbf{x}_{ref}(t) = [i_{ref}(t) \quad v_{ref}(t)]. \quad (3.7)$$

The first term in (3.6) penalizes the tracking error of the state variables, with

$$\mathbf{Q} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}, \quad (3.8)$$

as the penalty matrix. The diagonal entries of  $\mathbf{Q}$ , i.e.,  $\lambda_1$  and  $\lambda_2$ , are used to prioritize the different state variables, and we know that the main control objective is to minimize the current tracking error. Thus, the inductor current must be prioritized by penalizing the corresponding error more heavily than that of the active capacitor voltage. To achieve this,  $\lambda_1 > \lambda_2$ . Implying that a larger value of  $\lambda_1$  which corresponds to the current tracking error, is chosen.

The second term in (3.6) has

$$\Delta u(l) = u(l) - u(l-1),$$

as the switching effort and  $\lambda_u$  as the weighting factor.

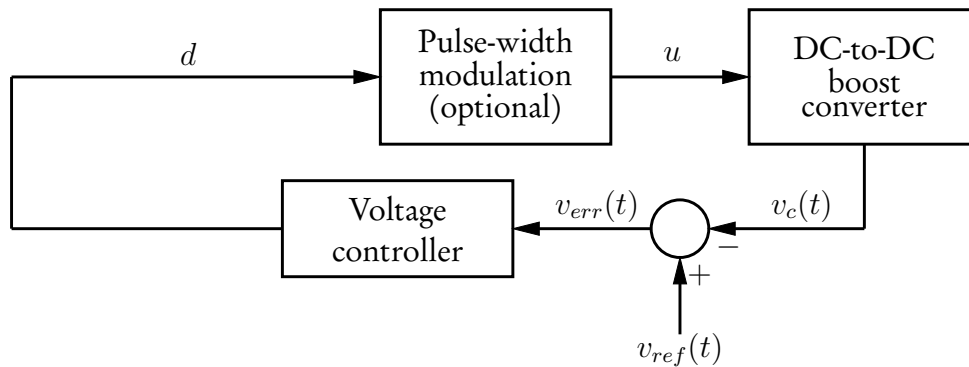
With the switching sequence of the switching states over the prediction horizon of  $N_p$ , introduced as  $\mathbf{U}(k) = [u(k) \quad u(k+1) \quad \dots \quad u(k+N_p-1)]^T$ , the optimization problem is stated as

$$\begin{aligned} \mathbf{U}_{opt}(k) &= \arg \min_{\mathbf{U}(k)} J \\ &\text{subject to (3.4),} \\ \mathbf{U}(k) &\in \{0, 1\}^{N_p}. \end{aligned} \quad (3.9)$$

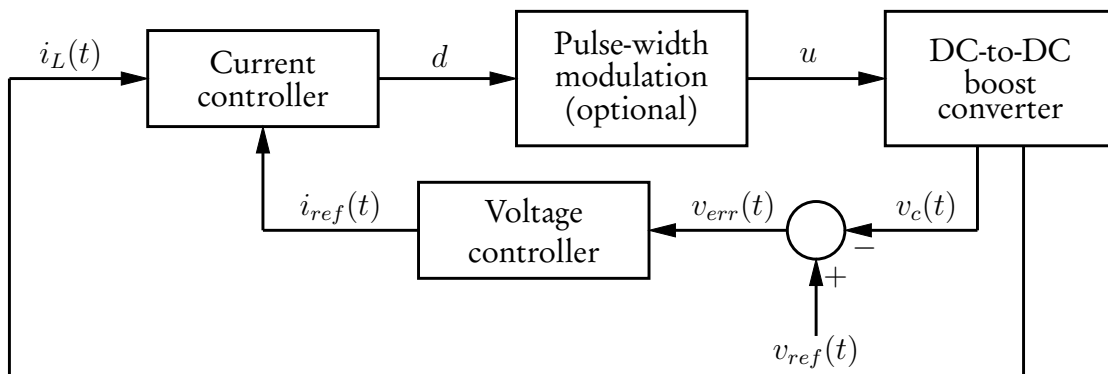
### 3.3 Optimal control of the boost converter

#### 3.3.1 Control of the active capacitor voltage

The most common control techniques for the closed-loop operation of a boost converter can be divided into two main groups, namely, voltage-mode control and current-mode control. Figure 3.4 shows the basic control loops.



(a) The voltage-mode control scheme.



(b) The current-mode control scheme.

Figure 3.4: Block diagrams of control loops for the boost converter. Adapted from [26].

With voltage-mode controllers, only a single control loop is used that directly controls the capacitor voltage in the absence of an intermediate current control loop as shown in Figure 3.4a. Typically, a modulator is used to indirectly manipulate the switching state  $u$ , by modifying the duty cycle  $d$ . The controller minimizes the voltage error  $v_{err}(t)$  between the measured capacitor voltage  $v_c(t)$  and its reference value  $v_{ref}(t)$ . If a linearized model of the boost converter is assumed, the control-to-output transfer function contains a zero on the right half-plane [26, 14]. This results in a reverse-response system behaviour during transients [26, 14]. Thus, with respect to the control input, i.e., the duty cycle, the control of capacitor voltage becomes difficult since it relates to a second-order system with a non-minimum phase behaviour [26].

The non-minimum phase behaviour is better explained with an example; upon increasing the voltage across the capacitor, the duty cycle must also increase. However, initially, the capacitor voltage drops before it actually starts to rise again, implying that the sign of the gain is not always positive.<sup>1</sup>

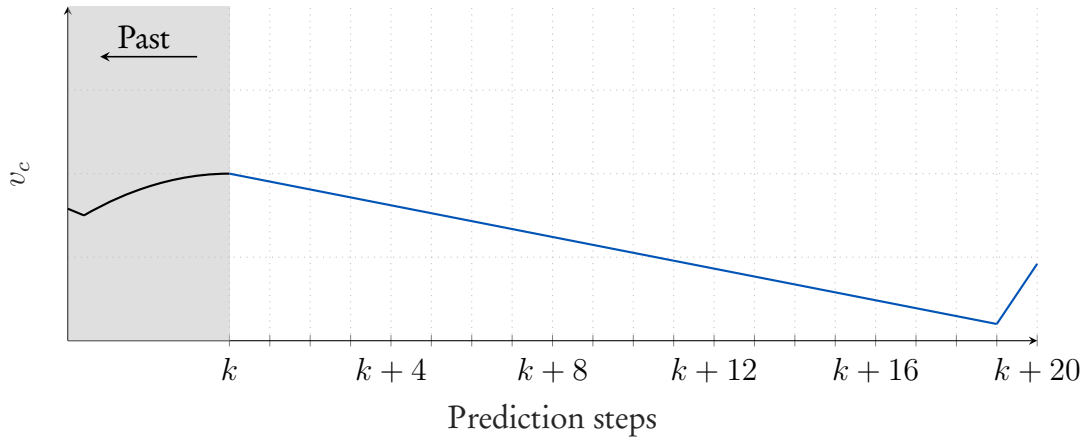
<sup>1</sup>The gain referred to here is from the duty cycle to the capacitor voltage.

On the other hand, current-mode controllers use two control loops. The control loops give rise to a cascaded control concept, which comprises of an inner and outer loop, as shown in Figure 3.4b. The outer loop regulates the measured capacitor voltage along its reference by manipulating the inductor current reference  $i_{ref}(t)$  in order to eliminate the voltage error. The inner loop regulates the measured inductor current  $i_L(t)$  along with its reference. Usually, current-mode controllers are used instead of voltage-mode controllers, since they exhibit a minimum phase behaviour with respect to the control action [26]. They relate to a first-order system.

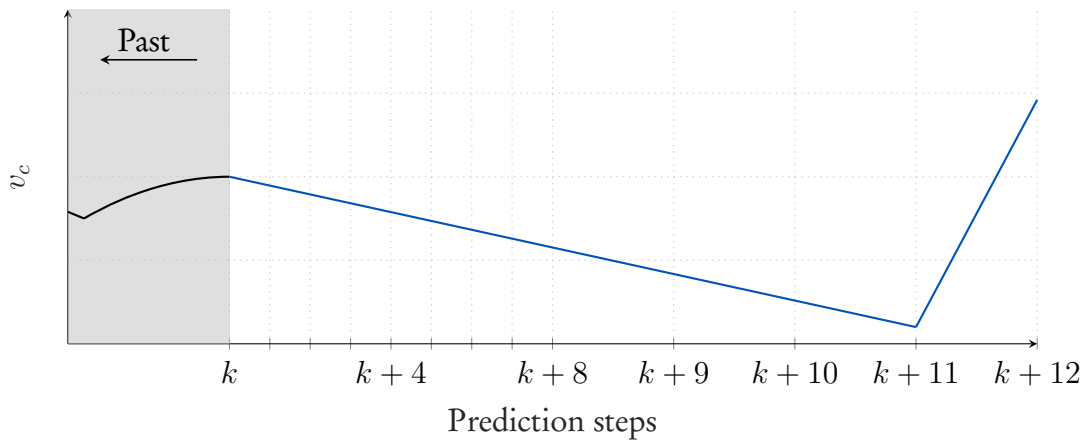
With that being said, the primary difficulty associated with boost converters becomes apparent when controlling their capacitor voltage without an intermediate current control loop [26].

The MPC based controller employed in this thesis achieves both voltage and current control with a single control loop, which is one of its main advantages. However, this implies that the active capacitor voltage is directly controlled and the system exhibits a non-minimum phase behaviour. To overcome the obstacle associated with direct voltage control, a sufficiently long prediction horizon  $N_p T_s$  is required. With a long prediction horizon, the controller can predict beyond the initial voltage drop when increasing the active capacitor voltage, and also ensures that any potential closed-loop stability issues are avoided [26]. Thanks to the move blocking strategy which was introduced in Section 2.5, a long prediction horizon can be achieved without a subsequent increase in the number of computations required to solve the optimization problem formulated in (3.9).

To illustrate the effectiveness of the move blocking strategy, consider Figure 3.5 where the given voltage reference for the capacitor voltage increases in a stepwise manner, at the time instant  $kT_s$ . The capacitor voltage  $v_c(t)$  is required to follow that reference. However, as already mentioned earlier, due to the non-minimum phase nature of the system, initially, the capacitor voltage tends to drop before it starts to increase. Thus, the controller must be able to predict the final voltage increase, and at the same time maintain closed-loop stability. In this example, a prediction horizon of  $N_p = 20$  is required to obtain a corresponding switching sequence that achieves this objective, refer to Figure 3.5a. The total length of the prediction horizon will be  $N_p T_s = 20T_s$ .



(a) Without move blocking.



(b) With move blocking.

Figure 3.5: Illustrative example of the effectiveness of the move blocking strategy [26]. Without move blocking, a prediction horizon of  $N_p = 20$  of equal time intervals is required. With move blocking, a prediction horizon of  $N_p = 12$  is sufficient to achieve the same closed-loop result obtained without move blocking, with  $N_1 = 8$ ,  $N_2 = 4$ , and  $n_s = 4$ . The total prediction length is  $24T_s$ .

Assuming that the optimization problem is solved via the enumeration of all possible switching sequences that can be assumed, then  $|\mathcal{U}|^{N_p}$  candidate solutions need to be examined.  $|\mathcal{U}|^{N_p}$  indicates the number of elements in  $\mathcal{U}$ . Refer to Section 2.3.6 for information regarding the exhaustive search method of solving the optimization problem.

With  $|\mathcal{U}| = 2$  and  $N_p = 20$ , results in  $2^{20} = 1\,048\,576$  candidate solutions. On the other hand, by employing the move blocking strategy and setting  $N_1 = 8$ ,  $N_2 = 4$  and  $n_s = 4$  (see Section 2.5.2 for the definition of these symbols), results in a prediction horizon of  $N_p = 12$ , and a total prediction length of  $24T_s$ , which is more than sufficient to achieve the desired goal.

Now, only  $2^{12} = 4\,096$  candidate solutions need to be examined. The computations required are reduced by three orders of magnitude, or by 99.6%. With the move blocking strategy, a much smaller number of prediction steps into the future are required to achieve the same closed-loop result obtained without move blocking, see Figure 3.5b.

In summary, with move blocking, we predict the first few instants very finely, then, we make the later predictions more coarse so we can see further into the future. Thus, we get the best of both worlds: a fine prediction and a long horizon in time, as explained in Section 2.5.2.

### 3.3.2 Control algorithm

In general, the optimization problem in (3.9) is computationally demanding. As explained in Section 2.3.7, its computational complexity increases exponentially by extending the prediction horizon. This indicates that the total number of candidate solutions for the problem under investigation increases when the prediction horizon is extended.

A relatively long prediction horizon is required to improve the performance of the boost converter, especially for the control of the active capacitor voltage, furthermore, to ensure the close tracking of the reference ripple current by the inductor current. In the previous section, the move blocking strategy was explained as a method of reducing the computations required while maintaining a sufficiently long prediction horizon. To further reduce the computations required, the branch-and-bound technique introduced in Section 2.4, can be used. Both the branch-and-bound technique, along with the move blocking strategy enable the implementation of the controller in real-time on an FPGA.

#### Initial upper bound cost algorithm

To warm-start the optimization process using the branch-and-bound technique, a good choice of the initial upper bound cost  $J_{ini}$  is required. The switching sequence  $\mathbf{U}_{ini}(k)$  used to compute the initial upper bound cost is obtained using the educated guess method as explained in Section 2.4.2. The code implementation for the initial upper bound cost calculation is shown in Algorithm 4.

---

#### Algorithm 4 Initial upper bound cost

---

```

1: function  $J_{ini} = \text{InitialCost}(\mathbf{x}_{meas}(k), u(k-1), \mathbf{U}_{opt}(k-1))$ 
2:    $\mathbf{x}(k) = \mathbf{x}_{meas}(k)$ 
3:    $J = 0$ 
4:    $j = 1$ 
5:   for  $j = 1$  to  $N_p$  do
6:     if  $j < k + N_1$  then
7:        $\mathbf{x}(j+1) = f_1(\mathbf{x}(j), U_{ini}(j))$ 
8:     else
9:        $\mathbf{x}(j+1) = f_2(\mathbf{x}(j), U_{ini}(j))$ 
10:    end if
11:     $\mathbf{x}_{err}(j+1) = \mathbf{x}_{ref}(j+1) - \mathbf{x}(j+1)$ 
12:    if  $j = 1$  then
13:       $J = J + \|\mathbf{x}_{err}(j+1)\|_Q^2 + \lambda_u \|\mathbf{u}(j-1) - \mathbf{U}_{ini}(j)\|_2^2$ 
14:    else
15:       $J = J + \|\mathbf{x}_{err}(j+1)\|_Q^2 + \lambda_u \|\Delta \mathbf{U}_{ini}(j)\|_2^2$ 
16:    end if
17:  end for
18:   $J_{ini} = J$ 
19: end function

```

---

In the algorithm, the function  $f$  represents the state updates of the discrete-time state-space model of the boost converter, given in (3.4). The subscripts of the function given as 1 and 2, correspond to the two different sampling intervals,  $T_s$  and  $T'_s$ , respectively, conforming to the move blocking strategy as explained in Section 2.5.2. The term  $\Delta \mathbf{U}_{ini}(j)$  in the initial cost calculation is given by  $\Delta \mathbf{U}_{ini}(j) = \mathbf{U}_{ini}(j-1) - \mathbf{U}_{ini}(j)$ .

### Non-recursive MPC algorithm

The algorithm presented in Section 2.4.3 that can be used to implement the branch-and-bound technique employs recursion (see Algorithm 2). The implementation of the branch-and-bound technique is complex, and on top of that, using recursion will come at a great expense with regards to the amount of resources used when implementing the controller in real-time on an FPGA. Therefore, solving the optimization problem recursively must, by all means, be avoided. In this thesis, the non-recursive branch-and-bound algorithm from [29] is adopted to implement the branch-and-bound technique, and it is modified to incorporate the move blocking strategy. The algorithm is shown in Algorithm 5 and will be explained thereafter.

---

#### Algorithm 5 Non-recursive MPC Algorithm

---

```

1: function  $u_{opt}(k) = \text{MPC}(\mathbf{x}(k), u(k-1), J_{ini})$ 
2:    $J_{opt}(k) = J_{ini}$ 
3:    $u_{opt}(k) = \emptyset$ 
4:    $j = 1$ 
5:   set each element in  $\mathbf{U}_{opt}(k) = 0$ 
6:   set each element in branch = 0
7:   set each element in  $\mathbf{J}_{cum} = 0$ 
8:   while solutionfound = 0 do
9:      $u_j = \text{branch}_j$ 
10:    if  $j < k + N_1$  then
11:       $\mathbf{x}(j+1) = f_1(\mathbf{x}(j), u(j))$ 
12:    else
13:       $\mathbf{x}(j+1) = f_2(\mathbf{x}(j), u(j))$ 
14:    end if
15:     $\mathbf{x}_{err}(j+1) = \mathbf{x}_{ref}(j+1) - \mathbf{x}(j+1)$ 
16:     $J = J_{cum_j} + \|\mathbf{x}_{err}(j+1)\|_Q^2 + \lambda_u \|\Delta u(j)\|_2^2$ 
17:    if  $J < J_{opt}(k)$  then
18:       $\mathbf{U}_{opt}(k) = u$ 
19:       $J_{opt}(k) = J$ 
20:       $\text{branch}_j ++$ 
21:    else
22:       $j ++$ 
23:       $J_{cum_j} = J$ 
24:    end if
25:    for  $q = N_p$  downto 2 do
26:      if  $\text{branch}_q > 1$  then
27:         $\text{branch}_q = 0$ 
28:         $j = q - 1$ 
29:         $\text{branch}_j ++$ 
30:      end if
31:    end for
32:    if  $\text{branch}_1 > 1$  then
33:      solutionfound = 1
34:    end if
35:  end while
36: end function

```

---

Both the recursive and non-recursive algorithm traverse the search tree in the same order as illustrated in an example in Section 2.4.3. However, the non-recursive algorithm, Algorithm 5, uses two different types of pointers to keep track of the explored branches. The first pointer, denoted by  $j$ , is known as the level pointer, which keeps track of what level in the search tree is currently being explored. The pointer has a range from 1 to  $n$  (where  $n = N_p$ ), i.e.,  $j = 1, 2, \dots, N_p$ . The second pointer, denoted by  $branch_j$ , is known as the branch pointer, where  $branch_j = 0, 1$ , referring to the admissible switch positions of the boost converter semiconductor switches. The branch pointer keeps track of what branch in the search tree is being investigated at a given level. The array **branch** stores  $N_p$  of such pointers.

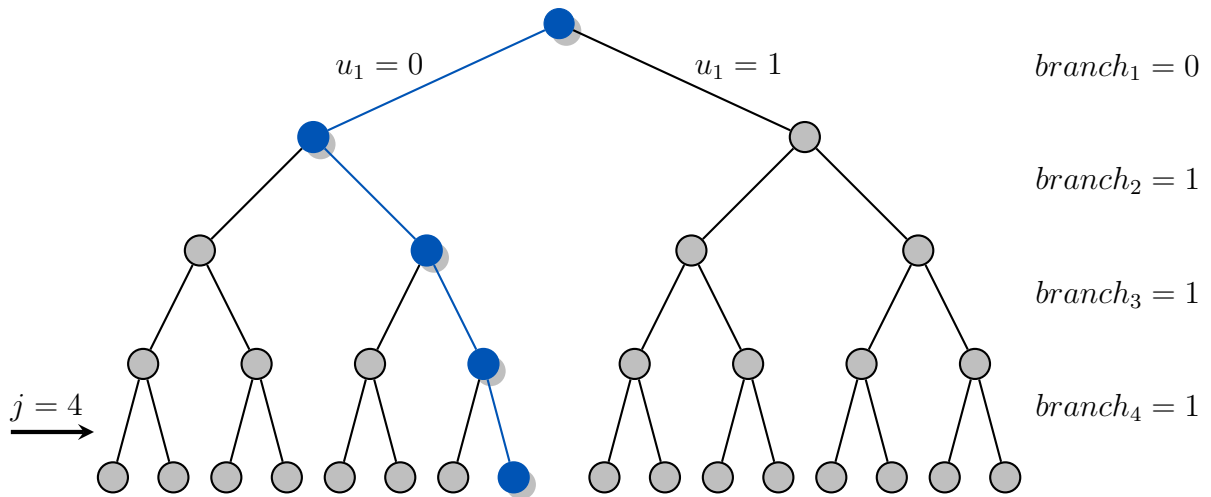
The major difference between the recursive and non-recursive algorithm is on how backtracking is performed. The non-recursive algorithm evaluates  $N_p - 1$  *if-statements* from line 25 to 31 to realize backtracking, whereas, with the recursive algorithm an element is popped out from a stack.<sup>2</sup>

The procedure for backtracking as realized in Algorithm 5, is illustrated in Figure 3.6. The way it works is explained as follows: first, recognize that the procedure evaluates the branch pointer from bottom to top, if a branch pointer is more than 1, it entails that the switch position does not form part of the set  $\mathcal{U} \in \{0, 1\}$ , thus, the branch pointer is set to 0. Thereafter, the current level is decreased by one and the next branch of the node is evaluated. Upon reaching the top, i.e., the first level, if the branch pointer  $branch_1 > 1$ , it entails that the entire search tree has been fully explored and the search algorithm terminates. A certificate of optimality is issued.

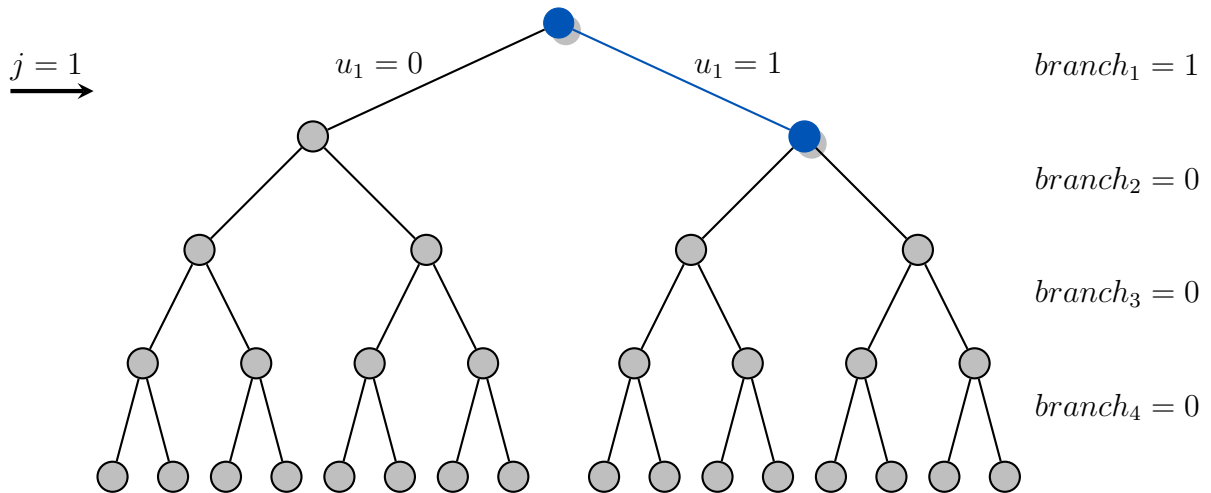
---

<sup>2</sup>Here, a stack is referring to the retrieving of data in an array as used in modern programming, in a “last in, first out” or LIFO order.





(a) In this example, the search algorithm has reached the leaf node, with the assembled switching sequence  $\mathbf{U}(k) = [0 \ 1 \ 1 \ 1]$ . Upon finishing the evaluation, the branch pointer at the bottom level is incremented to  $branch_4 = 2$ , however,  $2 \notin \mathcal{U}$ , therefore, backtracking commences.



(b) In this example, during backtracking, the branch pointer at the bottom-level  $j = 4$  is set back to  $branch_4 = 0$ , and the current level decreased to  $j = 3$ . Since the current branch has been explored, the branch pointer at that level is incremented to  $branch_3 = 2$  ( $\notin \mathcal{U}$ ), therefore it is set back to  $branch_3 = 0$ . After this, the current level decreased to  $j = 2$  and the branch pointer at the second level is incremented to  $branch_2 = 2$  ( $\notin \mathcal{U}$ ), thus, set back to  $branch_2 = 0$  and the current level is decreased to  $j = 1$ , the top-level. At the top level, the branch pointer is incremented to  $branch_1 = 1$ . Backtracking is completed and thus, the (partial) switching sequence assembled is now  $\mathbf{U}(k) = [1]$ .

Figure 3.6: Illustration of backtracking as realized by the non-recursive algorithm. The branch that is currently being explored is represented by the blue lines, which relates to the assembled switching sequence.

Once the switching sequence  $\mathbf{U}_{opt}(k) = [u_{opt}(k) \ u_{opt}(k+1) \ \dots \ u_{opt}(k+N_p-1)]^T$  is obtained from Algorithm 5, at time step  $k$ . Out of the sequence, only  $u_{opt}(k)$ , i.e., the first element, is applied to the boost converter. The rest of the switching sequence is discarded. The procedure is repeated based on new measurements at the next time step  $k+1$ . Figure 3.7 shows a block diagram of the proposed control system.

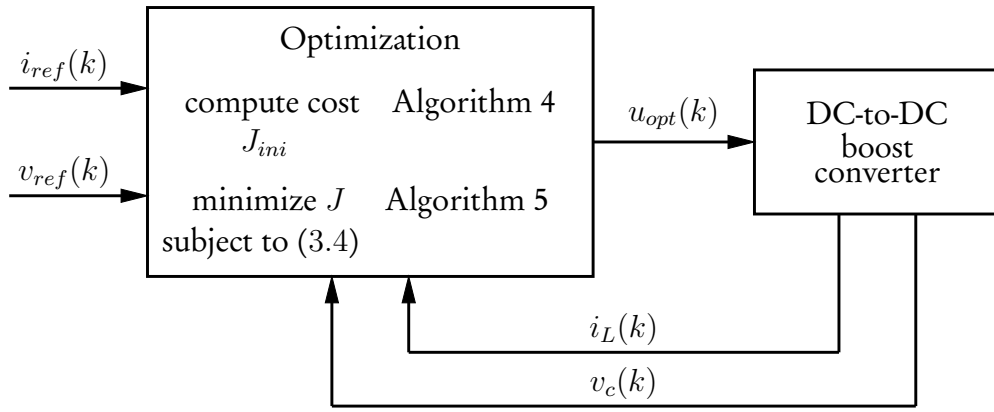


Figure 3.7: Block diagram of the proposed controller based on long-horizon direct MPC.

### 3.3.3 Generation of the reference ripple current

For ripple energy compensation to be possible, the reference ripple current  $i_{ref}(t)$  in (3.7), given to the controller, should have the correct magnitude and phase angle. In order to obtain an accurate reference ripple current, a band-pass filter can be used. A band-pass filter attenuates all the other harmonics except for the target harmonic, in this case, the second-order harmonic component. Therefore, it is well suited for this purpose. The transfer function of a second-order band-pass filter is

$$H_f(s) = \frac{H_0 \frac{\omega_f}{Q_f} s}{s^2 + \frac{\omega_f}{Q_f} s + \omega_f^2}, \quad (3.10)$$

where  $H_0$ ,  $\omega_f$  and  $Q_f$  denote the gain, the center frequency and the quality factor of the filter, respectively. The frequency-to-bandwidth ratio of the filter is denoted by the quality factor

$$Q_f = \frac{\omega_f}{\Delta\omega_f}, \quad (3.11)$$

with  $\Delta\omega_f$  as the bandwidth. The quality factor can be tuned to adjust the bandwidth of the filter. For this specific application, the filter has a gain of 1 (unity gain), and a centre frequency  $\omega_f = 2\pi f_r$ . Recall  $f_r = 2f_1$ . Figure 3.8 shows the frequency response of the band-pass filter.

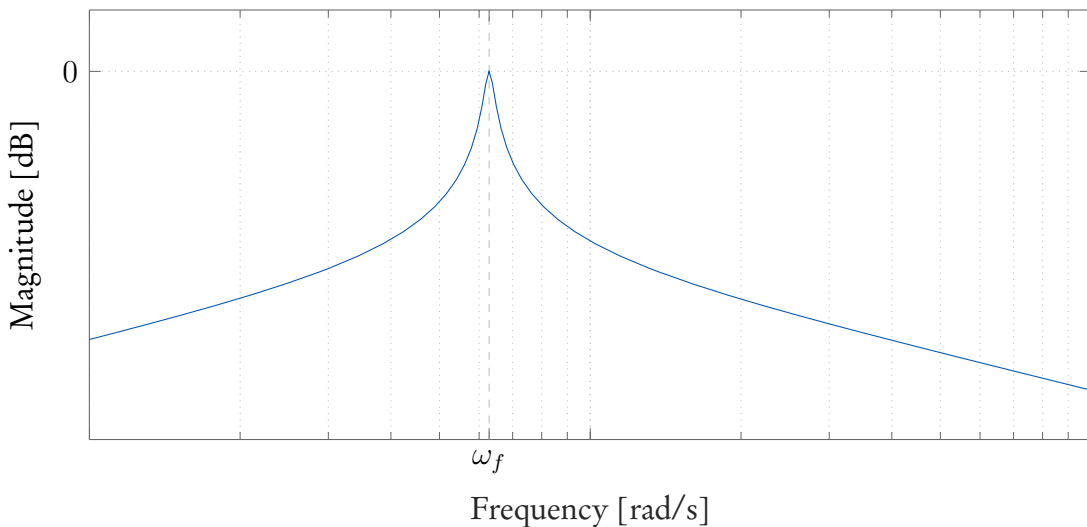


Figure 3.8: Frequency response of the band-pass filter.

The procedure of obtaining the reference ripple current is as follows; the rectified current  $i(t)$  that flows between the single-phase dc-to-ac converter and the boost converter, as shown in Figure 2.13, is passed through a band-pass filter. Figure 3.9 shows an illustration.

The rectified current consists of the dc component, the second-order harmonic component, and high-switching frequency components. Therefore, when passed through the filter, all the other harmonic components are attenuated, leaving only the second-order harmonic component, i.e., the ripple current  $i_r(t)$  in (2.32). Hence, at the output terminal of the filter, the ripple current is generated.

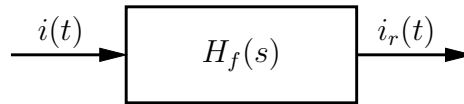


Figure 3.9: Illustration of the generation of the reference ripple current.

Note that the band-pass filter introduces a delay during the filtering process. The exact magnitude and phase angle of the ripple current is only obtained after  $i_r(t)$  has reached steady-state operation. Hence, the generated ripple current at the output terminal cannot be used in the controller directly from the filter. Instead, during steady-state operation, the magnitude and phase angle of  $i_r(t)$  is measured and used to re-calculate a suitable reference for the controller. The calculated ripple current is in the form

$$i_r(t) = I_r \cos(2\omega_1 t - \phi), \quad (3.12)$$

where  $I_r$  is the magnitude of the ripple current and  $\phi$  is the phase angle, as already mentioned in Section 2.11.

It is extremely important to note that the reference ripple current fed to the controller be  $180^\circ$  out of phase with the second-order harmonic component of  $i(t)$ . This allows for ripple energy compensation to be possible. Thus, the calculated reference ripple current

$$i_{ref}(t) = I_r \cos(2\omega_1 t - (\phi - 180^\circ)), \quad (3.13)$$

will be used in the controller.

### 3.4 Summary

In this chapter, the principles and concepts of MPC introduced in Chapter 2 were used. The modelling of the boost converter and the formulation of the optimal control law for the converter were presented. Due to the non-minimum phase behaviour that boost converters exhibit, controlling their capacitor voltage in the absence of an intermediate current control loop is challenging. In this thesis, the proposed MPC strategy directly controls the active capacitor voltage of the boost converter; thus, the move blocking strategy was used to address the associated problem. To further improve the performance of the system, long prediction horizons are required. Therefore, a non-recursive MPC algorithm that incorporates both a branch-and-bound technique and the move blocking strategy to implement long-horizon direct MPC, in an efficient manner on a low-cost FPGA, was formulated.

# Chapter 4

## Performance evaluation of long prediction horizons

### 4.1 Introduction

One crucial detail that the reader should take note of is that second-order harmonic currents generated on the dc side of a single-phase dc-to-ac converter are not produced only by a grid-connected system. A stand-alone system can also generate the same ripple current through the dc bus. In this chapter, simulations of both a grid-connected system and a stand-alone system are presented. The simulations are performed to verify the effectiveness of the control algorithms presented in Chapter 3, used to reduce the ripple current flowing through the battery to an acceptable value. Firstly, the simulation results obtained from the steady-state operation of both systems are presented, and the performance of each system is evaluated for selected prediction horizons, with  $N_p = 10$ , as the longest prediction horizon considered. Secondly, the response time of the controller during transients for a prediction horizon of  $N_p = 10$  is investigated. A similar response of the controller to transients would be expected for a grid-connected system, therefore, the simulations are only performed on the stand-alone system via a reference step in the load current. Finally, the chapter concludes with a summary.

### 4.2 Procedure for performing simulations

The efficacy of ripple energy compensation in a single-phase dc-to-ac converter using a boost converter with long-horizon MPC as the control method is verified through simulations in MATLAB® Simulink. The simulations are carried out for both a grid-connected system and a small-scale stand-alone system. For both systems, PWM with unipolar switching is used for the control of the single-phase dc-to-ac converter.<sup>1</sup> The switching frequency for the converter is set to  $f_c = 20$  kHz. The boost converter is controlled using long-horizon direct MPC.<sup>2</sup> Unless stated otherwise, the sampling interval of the model predictive controller is chosen as  $T_s = 25$   $\mu$ s. As mentioned in Section 2.3.4, the switching frequency for direct MPC is variable and it is adjusted using the weighting factor  $\lambda_u$ . To perform a thorough analysis of the benefit of using long horizons,  $\lambda_u$  must be adjusted to attain the desired average switching frequency for all the horizons considered. The chosen switching frequency  $f_{sw} = 16$  kHz.

<sup>1</sup>For convenience, the reader is referred to Section 2.1.3, for an in-depth study on PWM. Note that the pulse-width modulator used throughout in this thesis is an open-loop controller.

<sup>2</sup>The reader is referred to Chapter 3 for information on the control of a boost converter.

To generate the reference ripple current as discussed in Section 3.3.3, a discrete transfer function block that implements the Z-transform transfer function of the band-pass filter is used. A narrow bandwidth of 25 Hz is chosen, and as already mentioned earlier, the gain is set to 1. The center frequency of the filter is the same as that of the target harmonic required. In this case, the target harmonic  $f_r = 100$  Hz, since the fundamental frequency  $f_1 = 50$  Hz.

## 4.3 Steady-state operation

### 4.3.1 Grid-connected system

The parameters of the grid-connected system depicted in Figure 2.13, are given in Table 4.1.

Table 4.1: Grid-connected system simulation parameters.

Parameter	Symbol	SI value
Nominal line voltage	$V_{ac}$	230 V
Nominal dc voltage	$V_{dc}$	400 V
Dc-link capacitance	$C'_{dc}$	600 $\mu\text{F}$
Boost converter active capacitance	$C$	800 $\mu\text{F}$
Boost converter inductance	$L$	2.2 mH
Filter inductance	$L_{ac}$	2.2 mH
Power rating	$P_o$	18 kW
Power factor	$PF$	0.98

### Capacitance requirements

First and foremost, consider the case when the conventional method that utilizes a passive dc-link capacitor is used for ripple energy compensation, instead of using an active capacitor.<sup>3</sup> Considering a system with the rated parameters provided in Table 4.1, by using (2.23), the calculated minimum capacitance of the dc-link capacitor required will be  $C_{dc} = 17.9$  mF. This capacitance value results from the chosen maximum allowed voltage ripple component  $\Delta V_{dc}$  across the dc-link capacitor, which is 2 % of the nominal dc supply voltage.

Conversely, when using the active capacitor for ripple energy compensation, it is apparent that the capacitance requirements are significantly reduced. Using (2.31), with the average voltage of the active capacitor chosen as  $V_c = 700$  V, the minimum capacitance required is obtained as 233.9  $\mu\text{F}$ . Recall that for a boost converter, the instantaneous voltage of the capacitor must be higher than the dc bus voltage (see Section 2.2), hence the choice of the average voltage. This allows the active capacitor voltage to swing above the dc bus voltage.

During simulations, the capacitance of the active capacitor is adjusted in order to reduce the ripple current significantly. Finally an active capacitor  $C = 800$   $\mu\text{F}$  is chosen. A small dc-link capacitor  $C'_{dc} = 600$   $\mu\text{F}$  is used during the simulations. As mentioned before,  $C'_{dc}$  is used only to filter the high frequency harmonics caused by the switching of the single-phase dc-to-ac converter. Thus, the capacitance requirements for ripple energy compensation are significantly reduced by approximately 95.5 %.

<sup>3</sup>Refer to Section 2.1.5 for information on ripple energy compensation using a passive dc-link capacitor.

### Simulation waveforms

The grid-connected system with the parameters given in Table 4.1 is simulated and the rectified current  $i(t)$  is obtained. Figure 4.1 shows the simulation waveform.

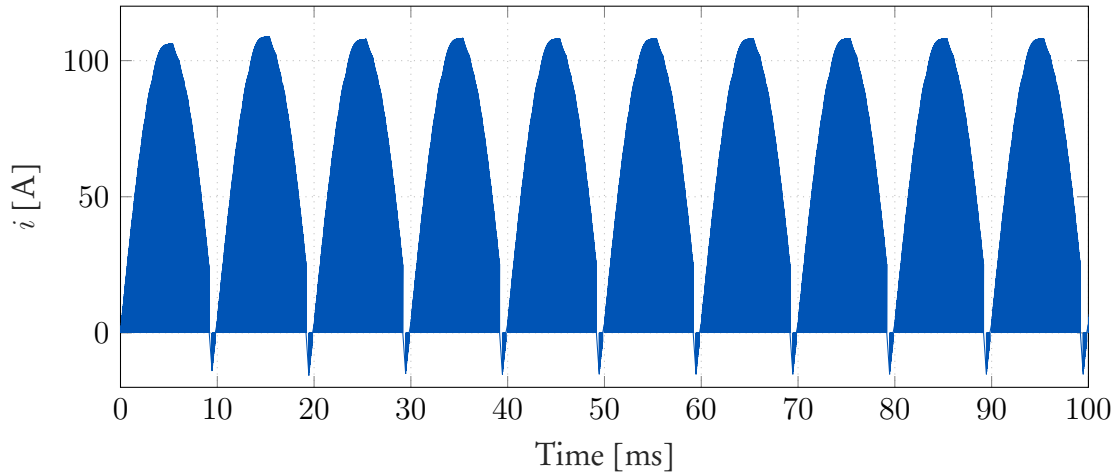


Figure 4.1: Simulation waveform of the rectified current  $i(t)$  obtained on the dc side of a PWM controlled grid-connected single-phase dc-to-ac converter.

The rectified current  $i(t)$  is passed through a band-pass filter during the simulation of the single-phase dc-to-ac converter. The ripple current in (3.12), i.e., the second-order harmonic component

$$i_{100\text{Hz}}(t) = 48.73 \cos(2\omega_1 t + 283.6^\circ),$$

is obtained at the output of the filter, in steady-state operation. The magnitude and phase angle of the second-order harmonic component is obtained. As already mentioned, the reference ripple current used in the model predictive controller has to be  $180^\circ$  out of phase with second-order harmonic component for ripple energy compensation to be possible. Therefore, the reference ripple current used in the controller

$$i_{ref}(t) = 48.73 \cos(2\omega_1 t + 103.6^\circ).$$

The reference voltage  $v_{ref}(t)$  of the active capacitor is calculated using (2.28). The coefficient value is set to  $k = 3.52$ , which is adjusted such that the instantaneous voltage of the active capacitor is always higher than the dc bus voltage. The phase angle of the reference voltage is determined by the phase angle of the reference ripple current. In general, the current through a capacitor is phase-shifted by  $90^\circ$ . Since the current that flows through the active capacitor is the ripple current, the reference voltage of the active capacitor should lag the reference ripple current by  $90^\circ$ . As a result, the calculated voltage reference of the active capacitor is given by

$$v_{ref}(t) = \sqrt{36622.08 - 10404 \cos(2\omega_1 t + 193.6^\circ)}.$$

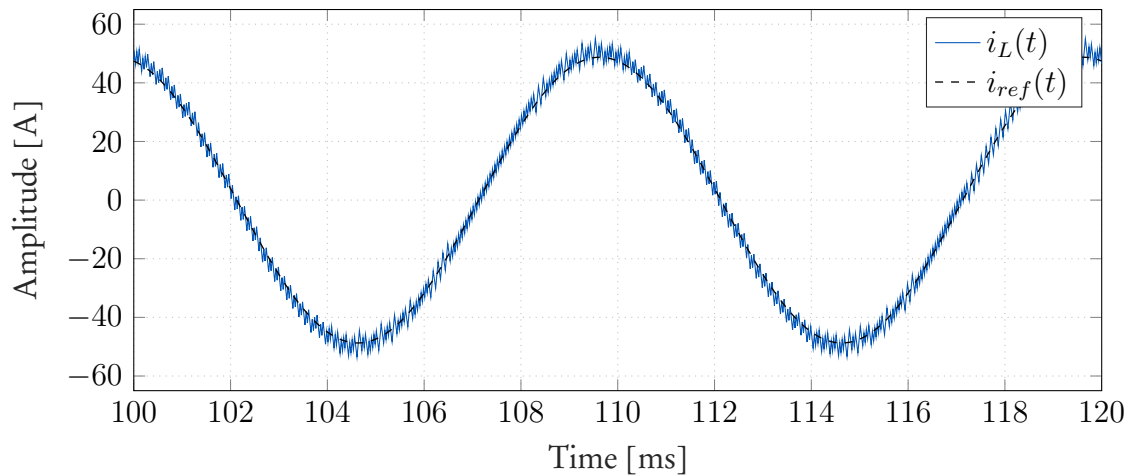
Both the reference ripple current  $i_{ref}(t)$  and the voltage reference  $v_{ref}(t)$  are given as entries to the reference vector  $\mathbf{x}_{ref}(t)$  in (3.7), for the implementation of the controller. Furthermore, the parameters conforming to the move blocking strategy are set to  $N_1 = 6$ ,  $N_2 = 4$  and  $n_s = 4$ , resulting in a prediction horizon of  $N_p = 10$ . This is the longest horizon considered. The move blocking strategy suggests that the first six steps of the prediction horizon be finely

sampled at  $T_s = 25 \mu\text{s}$ , and the last four steps of the prediction horizon sampled at  $T'_s = 100 \mu\text{s}$  (see Section 2.5.1). The penalty matrix and the scalar weighting factor in the cost function are chosen through trial-and-error as

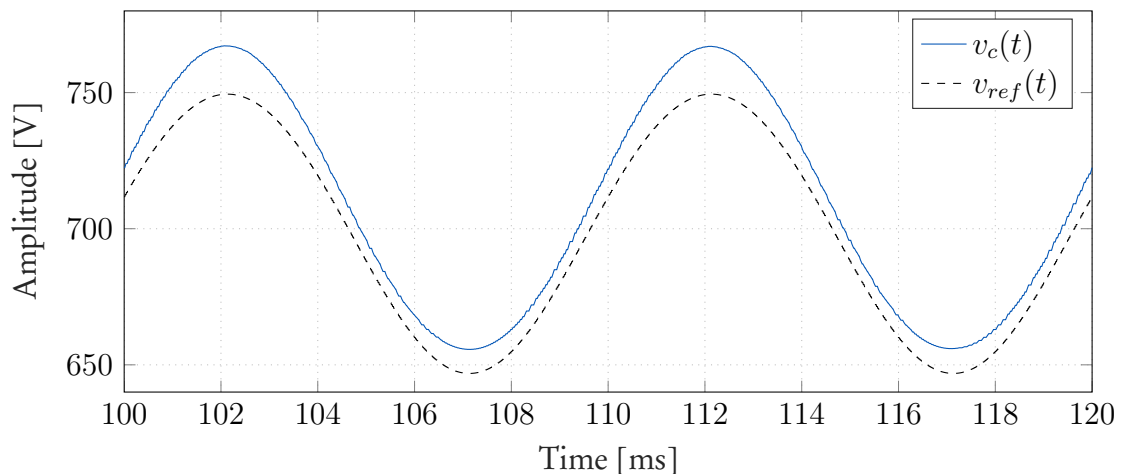
$$\mathbf{Q} = \text{diag}(90, 85) \text{ and } \lambda_2 = 1500,$$

respectively, with  $\lambda_u$  adjusted to attain an average switching frequency of  $f_{sw} \approx 16 \text{ kHz}$ .

The steady-state performance of the controller is shown in Figure 4.2. The reference signals are denoted by dashed lines, whereas solid lines denote the actual output signals of the boost converter.



(a) Reference tracking of the inductor current  $i_L(t)$ .



(b) Reference tracking of the active capacitor voltage  $v_c(t)$ .

Figure 4.2: Simulation waveforms of the inductor current and capacitor voltage of the boost converter for  $N_p = 10$ . The reference signals for the current and voltage,  $i_{ref}(t)$  and  $v_{ref}(t)$ , respectively, are denoted by the black dashed lines.

It should be apparent that the primary control objective of the model predictive controller is achieved. Figure 4.2a shows that the dashed lines are concealed by the solid lines. This implies that inductor current  $i_L(t)$  tracks the given reference ripple current  $i_{ref}(t)$  accurately. However, the active capacitor voltage  $v_c(t)$  does not accurately track the given time-varying voltage reference  $v_{ref}(t)$ . An offset exists between the two voltage signals, as shown in Figure 4.2b.

As mentioned in Section 3.2, the current tracking error is prioritized by penalizing it more heavily than the voltage tracking error. This ensures that all the ripple current flowing through the battery is diverted into the boost converter to the active capacitor via the inductor. In another simulation, when the voltage tracking error is penalized more heavily than the current tracking error. It is observed that the active capacitor voltage closely tracked the voltage reference. Nonetheless, the inductor current became too sluggish to follow the reference ripple current. This means the reference signals of the inductor current and active capacitor are contradicting each other. The reason for this contradiction could be attributed to the over-estimation of the importance of controlling either one of the system variables with respect to the other, suggesting that both the inductor current and active capacitor voltage control could be equally important. Also, the inductor current and active capacitor voltage are different in nature, thus, coupling effects between the two system variables could exist [30].

Another reason could be associated with the initial assumption made in (2.27). It suggests that all of the ripple power goes to the active capacitor. Thus, during positive ripple current flow through the boost converter, the current  $i_c(t)$  flowing through the active capacitor is equal to the ripple current. Implying that all of the ripple current charges the active capacitor. This assumption is used to derive the voltage reference in (2.28), which is used in the model predictive controller for the control of the active capacitor voltage.

However, due to the switch  $S_1$  between the inductor  $L$  and the active capacitor  $C$  of the boost converter, not all of the ripple current charges the active capacitor during the charging mode. When the control input  $u = 0$ , and the switch  $S_1$  is off, the ripple current flows through switch  $S_2$ . This is shown in Figure 2.14. From this, it indicates that the switching transitions of  $S_1$  and  $S_2$  during the charging and discharging modes of the active capacitor, need to be taken into account to derive a better reference for the active capacitor voltage.

Figure 4.3 shows the battery current  $i_b(t)$ , as labeled in Figure 2.13. The battery current is simulated with the initial voltages of the capacitors,  $C'_{dc}$  and  $C$ , set to the dc supply voltage of 400 V, to avoid the switch-on surge current when the boost converter is turned on.<sup>4</sup>

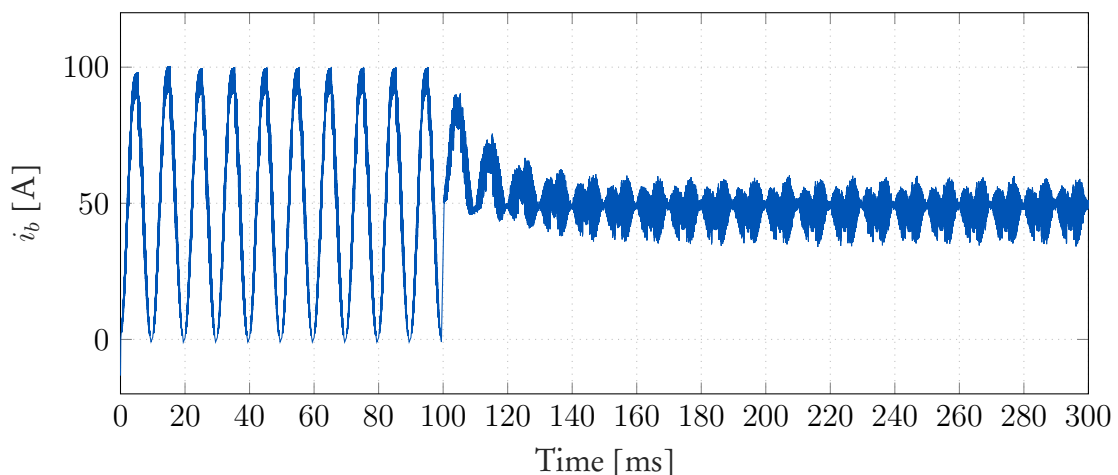


Figure 4.3: Simulation waveform of the battery current  $i_b(t)$  through the battery. For the time between  $t = 0$  and  $t = 100$  ms the boost converter is turned off, and for the time between  $t = 100$  ms and  $t = 300$  ms, the boost converter is turned on. This indicates the time before and after ripple energy compensation commences.

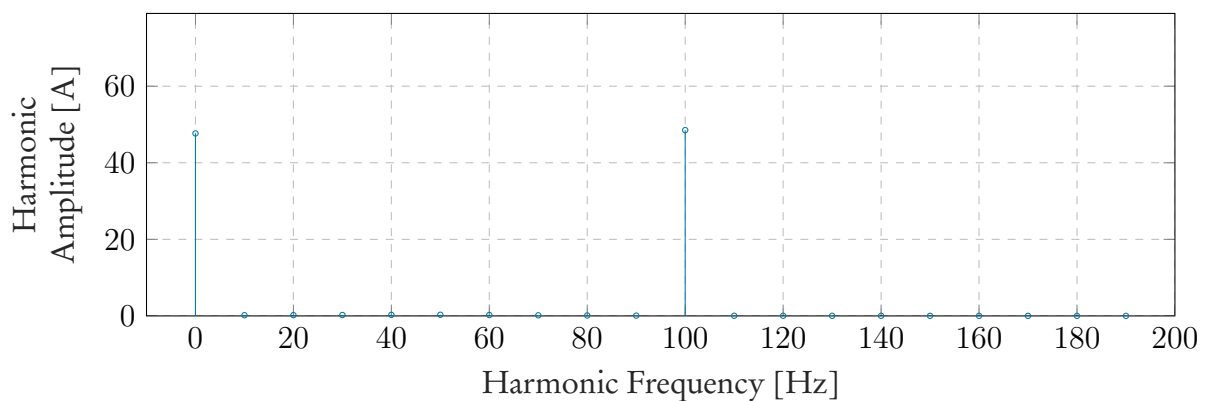
<sup>4</sup>In a physical practical circuit, a soft-start circuit is used to gradually increase the capacitor voltage to the potential of the dc voltage supply. This avoids an inrush current from causing a large current spike on the dc bus and thus, charging the capacitor too rapidly before the current goes into a steady-state operation.



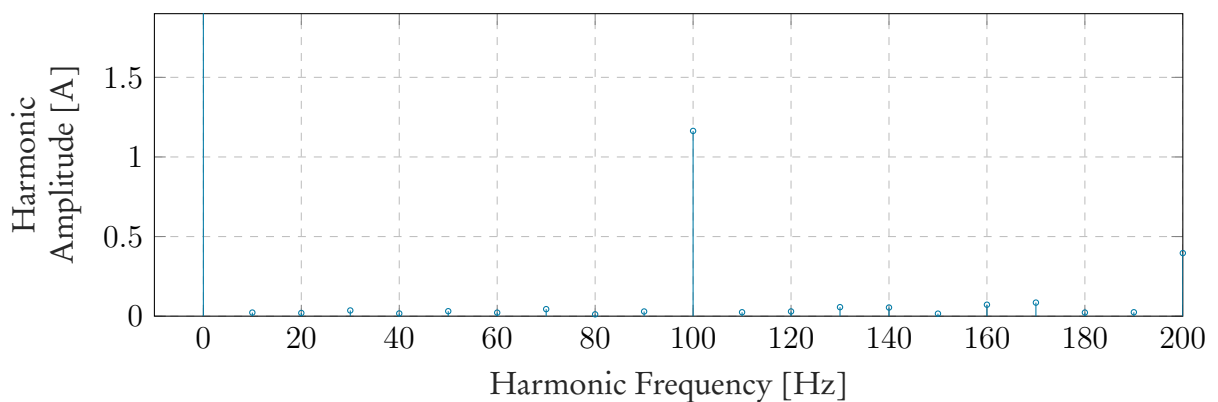
The simulation waveform of the battery current demonstrates ripple current reduction. From the time between  $t = 0$  and  $t = 100$  ms, no control input is applied to the semiconductor switches of the boost converter. Therefore, there is no ripple energy being compensated for. Immediately at  $t = 100$  ms, a control input  $u$  obtained from the non-recursive MPC algorithm, Algorithm 5, is applied to the boost converter semiconductor switches,  $S_1$  and  $S_2$ . Concurrently, the active capacitor starts to charge and discharge while compensating for the ripple energy. It is apparent that from  $t = 100$  ms going onwards, the amplitude of the ripple on the battery current is reduced significantly.

### Harmonic analysis

A fast Fourier transform (FFT) analysis is carried out to analyze the harmonic components contained in the current passing through the battery. Specifically, the second-order harmonic component. Figure 4.4 shows the harmonic spectrum of the battery current  $i_b(t)$  in Figure 4.3. First, the analysis is done for the time when the boost converter is turned off. The amplitude of the second-order harmonic component located at 100 Hz is found to be  $I_{100\text{Hz}} \approx 48.73$  A, as shown in Figure 4.4a. Thereafter, an analysis is done for the time when the boost converter is turned on. The resulting harmonic spectrum of the battery current is shown in Figure 4.4b. It can be seen that the amplitude of the second-order harmonic component has significantly reduced; the measured amplitude is  $I_{100\text{Hz}} \approx 1.18$  A.



(a) Current spectrum.



(b) Current spectrum.

Figure 4.4: Harmonic amplitude spectrum of the battery current  $i_b(t)$  in Figure 4.3 for the time before and after the boost converter is turned on for ripple energy compensation.

From the given results, it is apparent that the proposed control strategy successfully reduced the ripple current passing through the battery. The ripple current is not only reduced to the recommended 10 % of the nominal battery current, but it is reduced to about 2.4 % of the nominal battery current for a prediction horizon of  $N_p = 10$ .

### Performance evaluation of long horizons

The benefits of the long-prediction horizons on the proposed control strategy, for ripple energy compensation, are analyzed by considering a diverse set of weighting factors  $\lambda_u$  for different horizons. For each of the selected prediction horizon,  $\lambda_u$  is adjusted to try and match an average switching frequency of  $f_{sw} \approx 16$  kHz. The penalty matrix is kept at

$$Q = \text{diag}(90, 85)$$

same as for the previous simulations. Conforming to the move blocking strategy,  $n_s = 4$  is used throughout the simulations. Due to the non-minimum phase behaviour that the boost converter exhibits, unfortunately, for prediction horizons  $N_p < 3$ , the system is unstable and never reaches steady-state operation. This shows longer prediction horizons are required to solve the optimization problem successfully. Hence, only horizons  $N_p > 2$  are considered to attain closed-loop stability.

Table 4.2 shows the measured amplitude of the second-order harmonic component for the randomly selected prediction horizons. As already mentioned, the longest horizon considered is  $N_p = 10$ . It can be observed that longer prediction horizons (prediction horizons  $N_p > 3$ ) did not necessarily offer a significant decrease of the ripple current. For example, comparing the case of  $N_p = 3$  and  $N_p = 10$ , only a small difference of 0.78 A is observed between the amplitudes of ripple component. However, in general, a gradual decrease in the ripple current is noticed as the horizon is extended, except for the case of  $N_p = 6$ . This is attributed to the fact that the control system is unpredictable. It can therefore be concluded that, longer horizons tend to improve the performance of the system.

To analyze the effect of changing the number of prediction steps of the split-horizon, conforming to the move blocking strategy, the horizon of  $N_p = 7$  is used as an example. It can be seen that changing the number of steps for both the first and second part of the horizon,  $N_1$  and  $N_2$ , respectively, had a minor effect on the performance of the system. In general, it would be expected for the case with more prediction steps  $N_1$  that are finely sampled than the more course prediction steps  $N_2$  to have better performance. However, the fact that the switching frequency was not exactly the same between the cases makes the evaluation a bit biased. The same fact holds for all the prediction horizons considered for this analysis. This example indicates a general trend observed when changing the number of steps  $N_1$  and  $N_2$ .

Table 4.2: Comparison of the second-order harmonic component  $I_{100\text{ Hz}}$  at  $f_{sw} \approx 16$  kHz.

Horizon	$N_1 + N_2$	$\lambda_u$	$f_{sw}$ [kHz]	$I_{100\text{ Hz}}$ [A]
$N_p = 3$	2 + 1	1900	15.943	1.96
$N_p = 6$	4 + 2	1100	15.950	2.79
$N_p = 7$	5 + 2	1700	16.020	1.53
$N_p = 7$	4 + 3	1850	15.533	1.37
$N_p = 10$	6 + 4	1500	15.910	1.18

### 4.3.2 Stand-alone system

A small-scale stand-alone system can be used to evaluate the effectiveness of using long-horizon direct MPC for ripple energy compensation. It is important to note that the second-order harmonic component that appears on the dc side of a single-phase dc-to-ac converter is not only produced by a grid-connected system. The same characteristic applies to a stand-alone system. To get an understanding of how the second-order harmonic currents through the dc bus are produced in a stand-alone system, the reader is referred to Appendix B. In this section, the simulation results of a small-scale stand-alone system, i.e., a single-phase dc-to-ac converter with a  $RL$  load, denoted by  $R_g$  and  $L_g$ , are presented. The parameters are given in Table 4.3.

Table 4.3: Stand-alone system simulation parameters.

Parameter	Symbol	SI value
Nominal dc voltage	$V_{dc}$	48 V
Dc-link capacitance	$C'_{dc}$	2.0 mF
Boost converter active capacitance	$C$	2.1 mF
Boost converter inductance	$L$	800 $\mu$ H
Load inductance	$L_g$	800 $\mu$ H
Load resistance	$R_g$	0.8 $\Omega$
Power rating	$P_o$	1.2 kW

#### Capacitance requirements

Consider the case when a passive dc-link capacitor is used for ripple energy compensation. By using (2.23), with the system ratings in Table 4.3, and a voltage ripple component  $\Delta V_{dc}$  equal to 2% of the nominal dc supply voltage, the minimum capacitance required for the dc-link capacitor  $C_{dc} = 82.9$  mF. When utilizing the boost converter instead, by using (2.31), the minimum capacitance required for the active capacitor is  $C = 1.56$  mF. Through simulations, to significantly reduce the ripple current, the capacitance value is adjusted to  $C = 2.1$  mF. A smaller capacitor dc-link capacitor  $C'_{dc} = 2.0$  mF is used. Hence, the capacitance requirements for ripple energy compensation are reduced by 97.5%.

#### Simulation waveforms

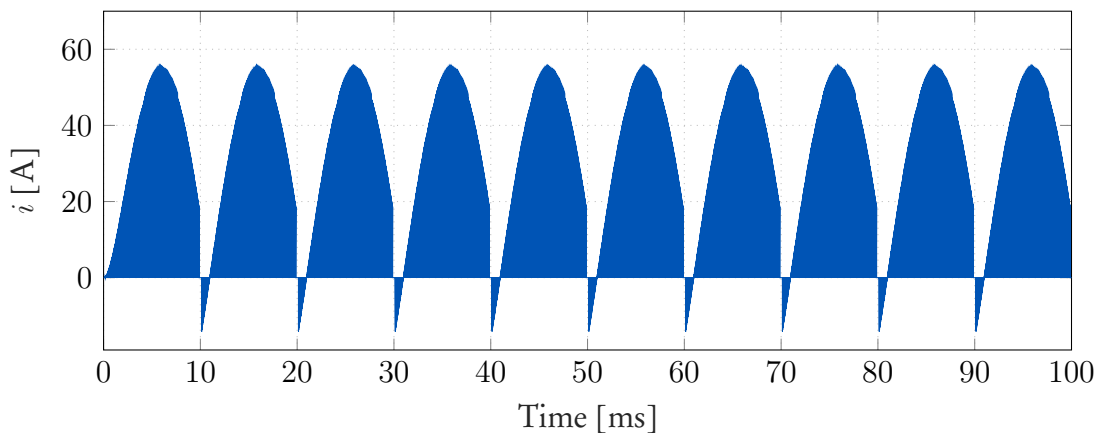


Figure 4.5: Simulation waveform of the rectified current  $i(t)$  obtained on the dc side of a PWM controlled single-phase dc-to-ac converter with an  $RL$  load.

The rectified current  $i(t)$  obtained from the stand-alone system, which is shown in Figure 4.5, is passed through a band-pass filter. The second-order harmonic component

$$i_{100\text{Hz}}(t) = 26.50 \cos(2\omega_1 t + 162.6^\circ),$$

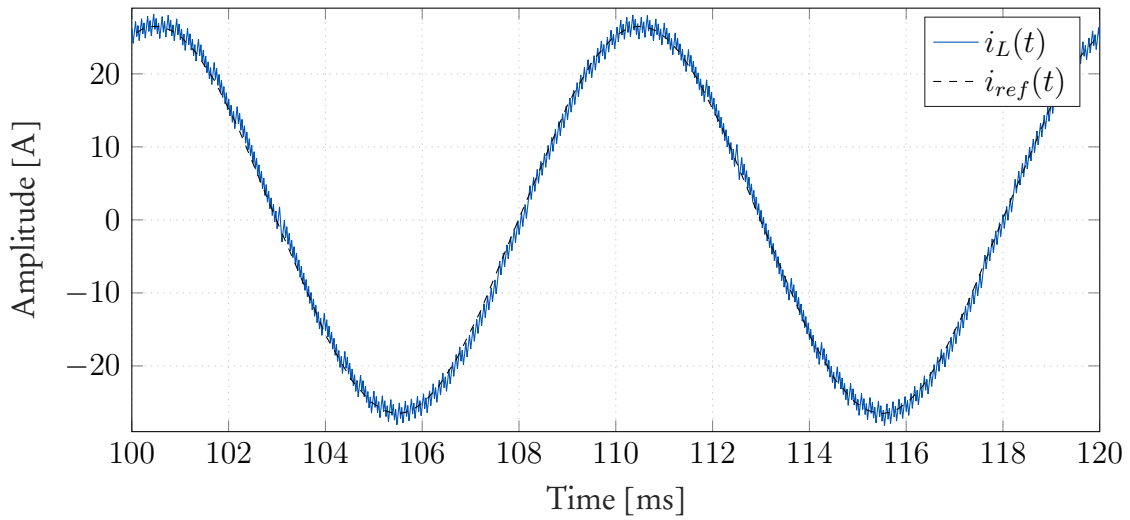
is obtained in steady-state operation. The reference ripple current used in the controller

$$i_{ref}(t) = 26.50 \cos(2\omega_1 t - 17.4^\circ).$$

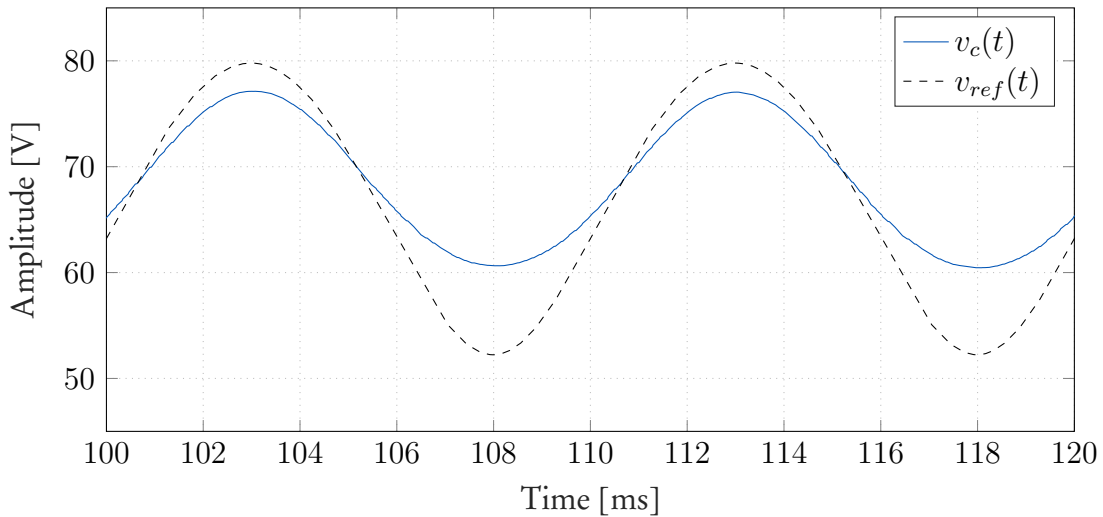
The reference voltage of the active capacitor is calculated using (2.28), and it is phase-shifted by  $90^\circ$  from the reference ripple current. It is given by

$$v_{ref}(t) = \sqrt{4547.3 - 18189 \cos(2\omega_1 t + 72.5^\circ)},$$

For the calculation, a coefficient value of  $k = 2.5$  is used, which is chosen such that the instantaneous voltage of the active capacitor is always higher than the dc bus voltage.



(a) Reference tracking of the inductor current  $i_L(t)$ .



(b) Reference tracking of the active capacitor voltage  $v_c(t)$ .

Figure 4.6: Simulation waveforms of the inductor current and capacitor voltage of the boost converter for  $N_p = 10$ . The reference signals for the current and voltage,  $i_{ref}(t)$  and  $v_{ref}(t)$ , respectively, are denoted by the black dashed lines.

The reference ripple current  $i_{ref}(t)$  and the voltage reference  $v_{ref}(t)$  are given as entries to the reference vector  $\mathbf{x}_{ref}(t)$  in (3.7), for the implementation of the controller. Figure 4.6 shows the simulations waveforms of the inductor current and the active capacitor voltage of the boost converter, along with their references, for  $N_p = 10$ . In accordance with the move blocking strategy, the horizon is split as follows:  $N_1 = 6$ ,  $N_2 = 4$ , and with  $n_s = 4$ . The penalty matrix and the scalar weighting factor in the cost function are chosen as

$$\mathbf{Q} = \text{diag}(250, 90) \quad \text{and} \quad \lambda_2 = 10,$$

respectively, with  $\lambda_u$  adjusted to obtain an average switching frequency of  $f_{sw} \approx 16$  kHz.

Figure 4.6a shows that the inductor current  $i_L(t)$  closely tracks the given reference ripple current  $i_{ref}(t)$ . Thus, the primary control objective is met. Figure 4.6b indicates the same difficulty when controlling the active capacitor voltage as in the grid-connected system. In this case the amplitude of the active capacitor voltage seems to be much less than the given voltage reference. Hence, a larger offset exists between the two voltage signals. It is believed that the same reasons mentioned in Section 4.3.1 attribute to the difficulty encountered when controlling the active capacitor voltage.

Figure 4.7 shows the battery current  $i_b(t)$  obtained from the simulation of the stand-alone system. The initial voltages of the capacitors  $C'_{dc}$  and  $C$  are initially set to the same potential as the dc supply voltage of 48 V. This is done to avoid a large current spike on the dc bus when the boost converter is turned on. For the time before  $t = 100$  ms, the boost converter is turned off. Promptly at  $t = 100$  ms going onwards, the boost converter is turned on. Evidently, the amplitude of the ripple on the battery current is significantly reduced upon turning on the boost converter, implying that the ripple current flowing through the battery is reduced.

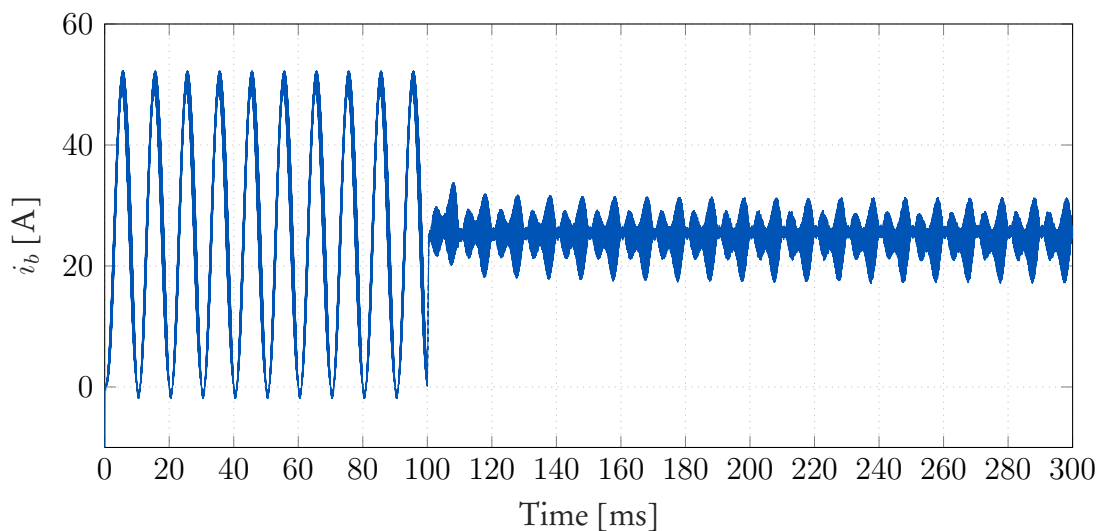
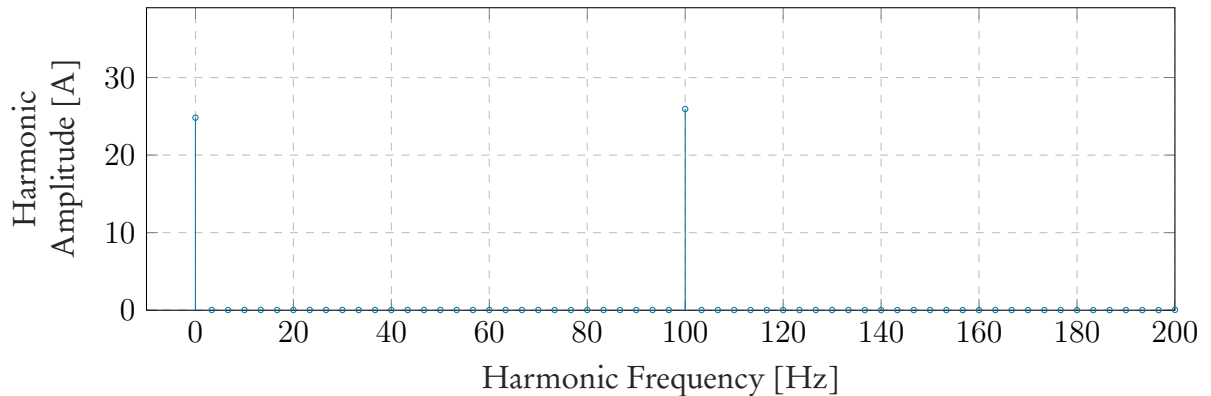


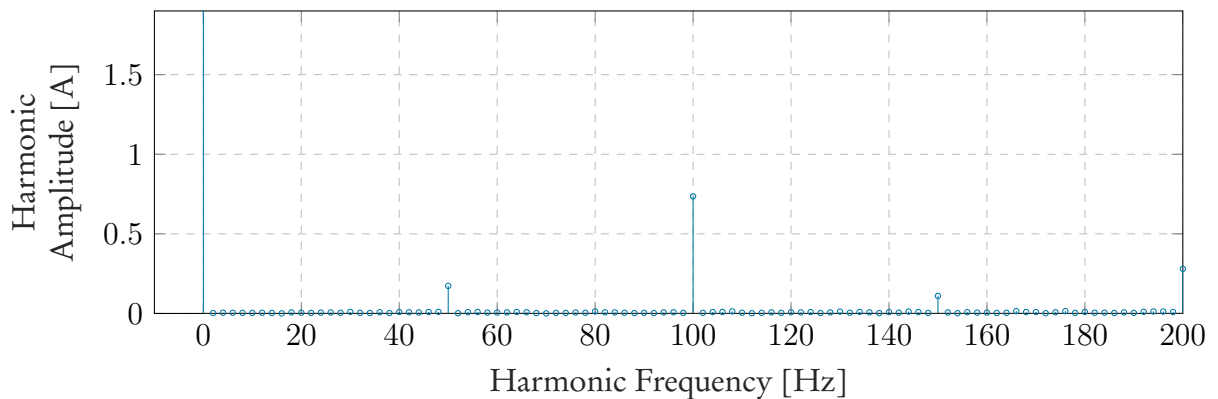
Figure 4.7: Simulation waveform of the current  $i_b(t)$  through the battery. The waveform shows the current for the time before and after the boost converter is turned on to compensate for the ripple energy. The time between  $t = 0$  and  $t = 100$  ms the boost converter is turned off, and from the time between  $t = 100$  ms and  $t = 300$  ms, the boost converter is turned on.

### Harmonic analysis

An FFT analysis of the battery current in Figure 4.7 is done. The analysis is done for both times, before and after the boost converter is turned on, to analyze the amplitude of the second-order harmonic component  $I_{100\text{Hz}}$ . Figure 4.8 shows the harmonic spectrum of the battery current. For the time when the boost converter is turned off,  $I_{100\text{Hz}} \approx 26.50\text{ A}$ , as shown in Figure 4.8a. After the boost converter is turned on,  $I_{100\text{Hz}} \approx 0.74\text{ A}$ , as shown in Figure 4.8b. The results show that the ripple current flowing through the battery is greatly reduced to about 2.8% of the nominal battery current.



(a) Current spectrum.



(b) Current spectrum.

Figure 4.8: Harmonic amplitude spectrum of the battery current  $i_b(t)$  in Figure 4.7 for the time before and after the boost converter is turned on to compensate for the ripple energy.

### Performance evaluation of long horizons

An analysis on the benefits of long-prediction horizons for ripple energy compensation in a stand-alone system is carried out for different horizons. For the chosen prediction horizons,  $\lambda_u$  is adjusted to achieve an average switching frequency of  $f_{sw} \approx 16\text{ kHz}$ . The penalty matrix

$$Q = \text{diag}(250, 90)$$

is kept the same throughout the simulations. Again, only the prediction horizons  $N_p > 2$  are considered since the system is unstable for horizons less than  $N_p = 3$ , due to the non-minimum phase behaviour that the boost converter exhibits. For the implementation of the move blocking strategy, the value  $n_s = 4$  is kept the same throughout the simulations.

In Table 4.4, a comparison of the amplitudes of the second-order harmonic component  $I_{100\text{Hz}}$  for the selected prediction horizons are shown. The longest horizon considered is  $N_p = 10$ . As it can be seen, longer prediction horizons improved the system performance, comparing  $N_p = 3$  and the successive horizons up to  $N_p = 10$ . It can also be seen that prediction horizons,  $N_p = 6$  and  $N_p = 7$ , had better performance than  $N_p = 10$ . The control system is unpredictable, therefore, such occurrences are to be expected. Also, the switching frequency of each horizon approximates the desired  $f_{sw} = 16\text{ kHz}$ , thus, making the analysis a bit unfair.

Using the prediction horizon of  $N_p = 7$  as an example, it can be seen that changing the number of prediction steps of the split-horizon had an effect on the performance of the system. By changing the values of  $N_1$  and  $N_2$ , for the first and second part of the horizon, respectively, the amplitude of the second-order harmonic component changed, however, the difference is rather small and considered insignificant. In this case the horizon with more  $N_1$  sampling instants that are finely sampled performed better than the one with more  $N_2$  sampling instants that are coarsely sampled, as would be expected. The difference can also be potentially attributed to the slight difference in the switching frequencies.

Table 4.4: Comparison of the second-order harmonic component  $I_{100\text{Hz}}$  at  $f_{sw} \approx 16\text{ kHz}$ .

Horizon	$N_1 + N_2$	$\lambda_u$	$f_{sw}$ [kHz]	$I_{100\text{Hz}}$ [A]
$N_p = 3$	2 + 1	10	17.385	1.21
$N_p = 6$	4 + 2	30	16.050	0.67
$N_p = 7$	5 + 2	27	16.101	0.70
$N_p = 7$	4 + 3	25	16.001	0.73
$N_p = 10$	6 + 4	10	16.176	0.74

## 4.4 Response during transients

MPC has the benefit of fast response time. However, the computational burden of solving the optimization problem during transients is more extensive compared with the steady-state operation, especially if long prediction horizons are involved.

The performance of the model predictive controller during transients is evaluated only on the stand-alone system, since a similar response of the controller to transients would be expected for a grid-connected system. The evaluation is conducted with a horizon of  $N_p = 10$ , as it is the longest horizon considered, with the highest computational burden. The prediction steps are set to  $N_1 = 6$  and  $N_2 = 4$ , with  $n_s = 4$ . The penalty matrix and the scalar weighting factor in the cost function are kept as

$$\mathbf{Q} = \text{diag}(250, 90) \quad \text{and} \quad \lambda_2 = 10,$$

same as the simulation of the stand-alone system in Section 4.3.2, with  $N_p = 10$ . The transient response is investigated for two cases: a step-up change and a step-down change in the reference signals of the load current  $i_g(t)$ .

### Step-up change in the load current

A step in the modulation index from  $\frac{m_a}{2}$  to  $m_a$  is introduced to the reference signals of the pulse-width modulator, with  $m_a = 0.96$ . Figure 4.9 shows that the load current  $i_g(t)$  increased at time  $t = 2\text{ s}$  from an amplitude of 27.39 A to 54.77 A.

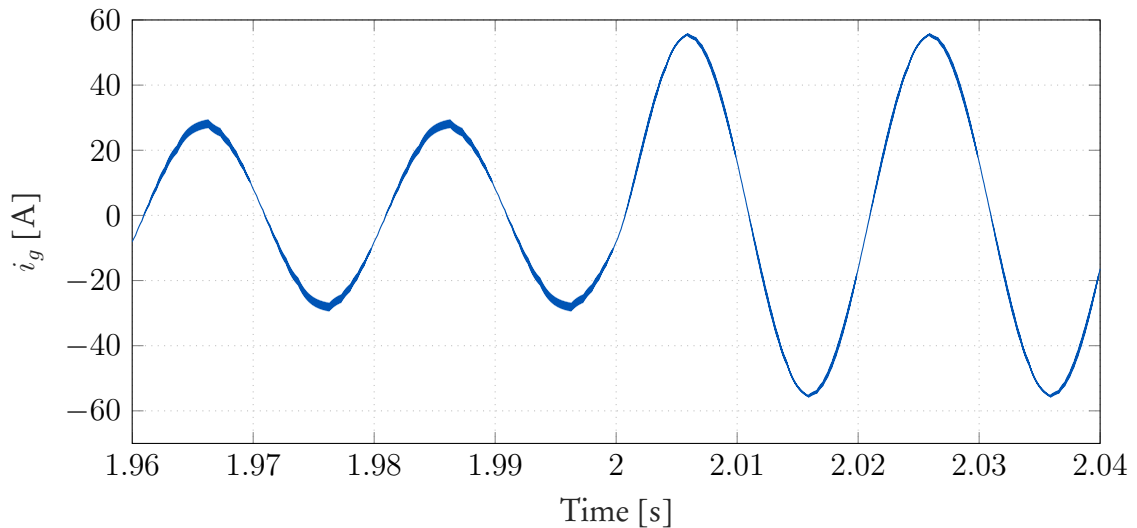


Figure 4.9: Simulation waveform of the load current  $i_g(t)$  of the stand-alone system with a step-up change at  $t = 2$  s.

The corresponding rectified current  $i(t)$  is passed through the band-pass filter to obtain the magnitude and phase angle of the second-order harmonic component. The accurate measurements of the magnitude and phase angle are obtained during a steady-state operation before and after the step change is implemented due to the delay caused by the filter. The reference ripple current used in the controller

$$i_{ref}(t) = 6.64 \cos(2\omega_1 t - 17.4^\circ),$$

and

$$i_{ref}(t) = 26.50 \cos(2\omega_1 t - 17.4^\circ),$$

for the time before and after the step change, respectively.

Figure 4.10 shows the inductor current  $i_L(t)$  of the boost converter when a step-up change is introduced at  $t = 2$  s. The reference ripple current  $i_{ref}(t)$ , as defined above, is also shown in Figure 4.10, and denoted by the black dashed lines. It can be seen that immediately when the step-up change is introduced at  $t = 2$  s, the inductor current instantly responds to the step change and oscillates for roughly 10 ms. At  $t = 2.01$  s, the inductor current promptly goes into a steady-state without any damping oscillations until it reaches a steady-state value. It appears as though there are two transients.

The transient response seen at  $t = 2.01$  s relates to an ideal case of a *critically damped* response, since there are no oscillations about the steady-state value. One could argue that this could be related to an overdamped response that also has no oscillations. However, an overdamped oscillation takes a long time until it reaches a steady-state value. In this case, there is almost zero settling time from when the transient occurs at  $t = 2.01$  s to the time when the inductor current goes into a steady-state, indicating a fast response time to a transient.

Since the instantaneous voltage of the active capacitor is supposed to be higher than the dc bus voltage, its reference voltage is kept as

$$v_{ref}(t) = \sqrt{4547.3 - 18189 \cos(2\omega_1 t + 72.5^\circ)},$$

the same as in the simulations of the stand-alone system in Section 4.3.2. Figure 4.11 shows the corresponding active capacitor voltage  $v_c(t)$  before and after the step-up change in applied at  $t = 2$  s.



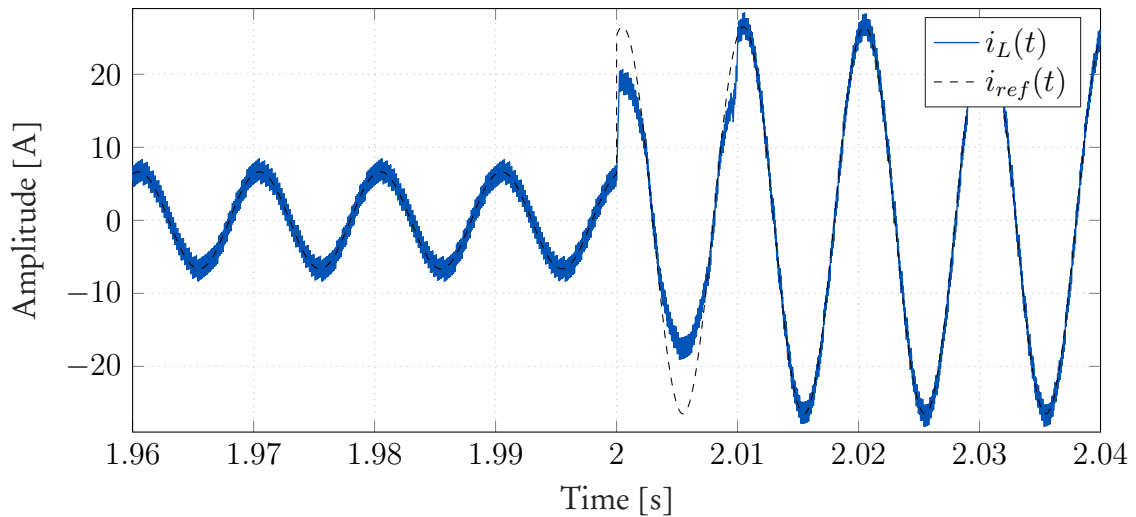


Figure 4.10: Simulation waveform of the inductor current  $i_L(t)$  for a step-up change in the load current at  $t = 2$  s. The reference ripple current  $i_{ref}(t)$  is denoted by the black dashed line.

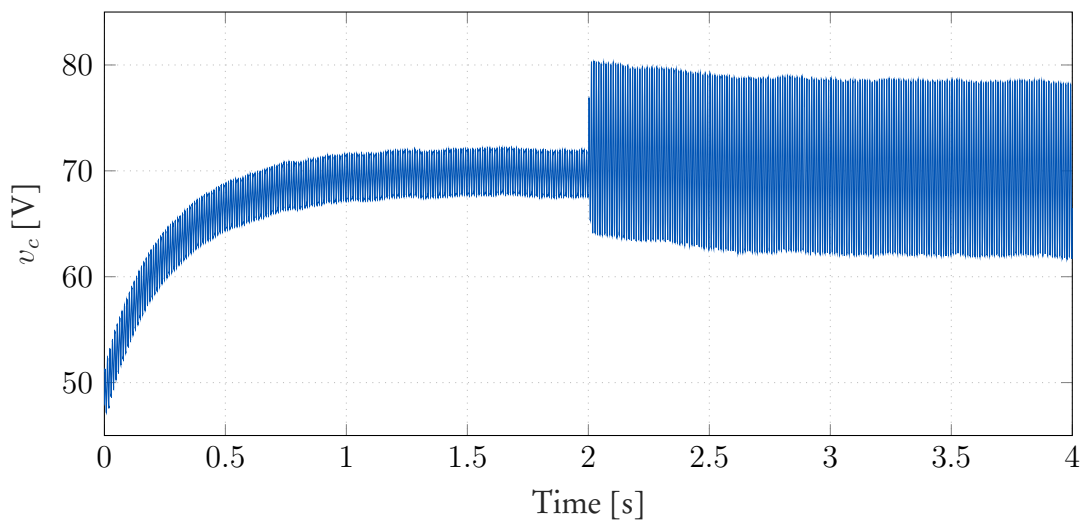


Figure 4.11: Simulation waveform of the active capacitor voltage  $v_c(t)$  after a step-up change in the load current at  $t = 2$  s.

In Figure 4.12 the battery current  $i_b(t)$  resulting from the step-up change is shown. It can be seen that when the step change is introduced at  $t = 2$  s, the battery current quickly ramps up and an overshoot in the current occurs for approximately 10 ms before another transient occurs at  $t = 2.01$  s. Again, it appears that there are two transients to a step-up change in its reference, similar to the response of the inductor current  $i_L(t)$ . Immediately at  $t = 2.01$  s, the battery current goes into a steady-state without any damping oscillations.

The amplitude of the second-order harmonic component is measured for both times before and after the step-up change as,  $I_{100\text{Hz}} = 0.37$  A and  $I_{100\text{Hz}} = 0.74$  A, from an initial value of,  $I_{100\text{Hz}} = 6.64$  A and  $I_{100\text{Hz}} = 26.50$  A, respectively.

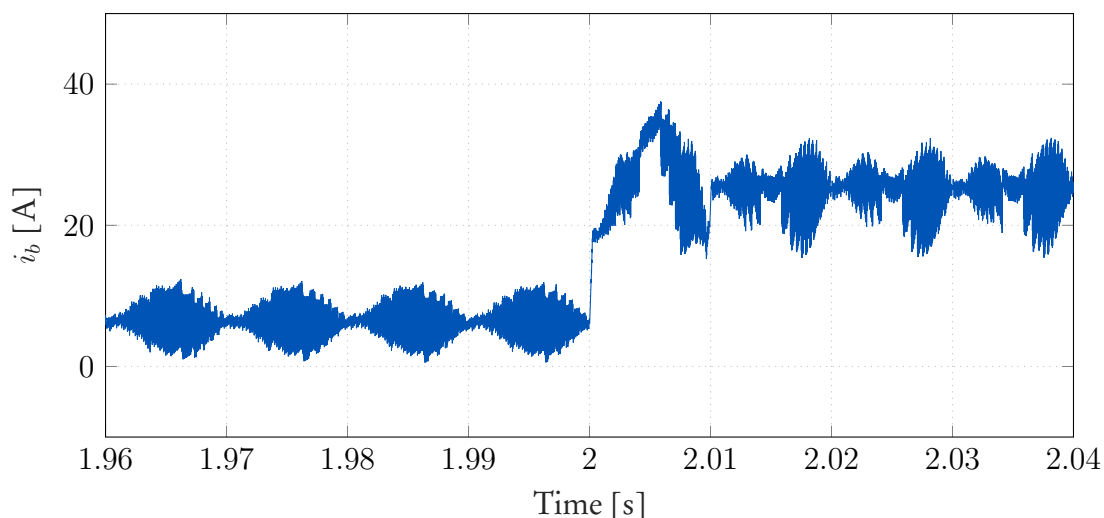


Figure 4.12: Simulation waveform of the battery current  $i_b(t)$  for a step-up change in the load current at  $t = 2$  s.

#### Step-down change in the load current

Conversely, a step in the modulation index from  $m_a$  to  $\frac{m_a}{2}$  is introduced to the reference signals of the pulse-width modulator, with  $m_a = 0.96$ . Figure 4.13 shows the load current  $i_g(t)$  decreased at time  $t = 2$  s from an amplitude of 54.77 A to 27.39 A.

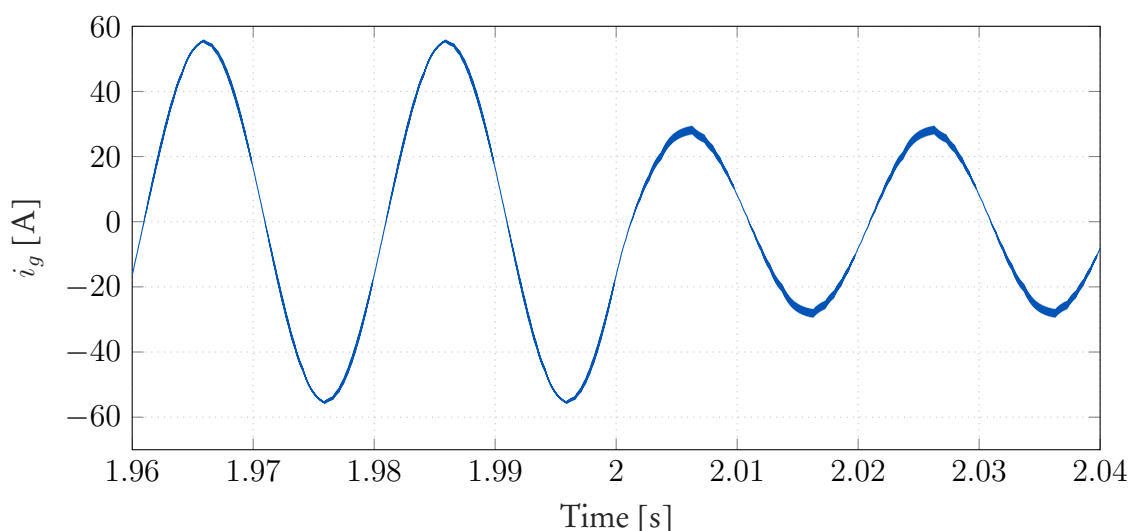


Figure 4.13: Simulation waveform of the load current  $i_g(t)$  of the stand-alone system with a step-down change in the load current at  $t = 2$  s.

As explained previously, for the step-up change in the load current, the same procedure of obtaining the reference ripple current without delay in calculating the reference is followed. The reference ripple current used in the controller

$$i_{ref}(t) = 26.50 \cos(2\omega_1 t - 17.4^\circ),$$

and

$$i_{ref}(t) = 6.64 \cos(2\omega_1 t - 17.4^\circ),$$

for the time before and after the step change, respectively. For the active capacitor voltage, the same voltage reference

$$v_{ref}(t) = \sqrt{4547.3 - 18189 \cos(2\omega_1 t + 72.5^\circ)},$$

is used. Figure 4.14 shows the inductor current  $i_L(t)$  when a step-down change is introduced at  $t = 2$  s and the black dashed line denotes the reference ripple current  $i_{ref}(t)$ . Similar to the step-up change, another transient occurs after 10 ms, at  $t = 2.01$  s, and almost instantly the inductor current goes into a steady-state.

Figure 4.15 shows the active capacitor voltage corresponding to the step-down change at  $t = 2$  s.

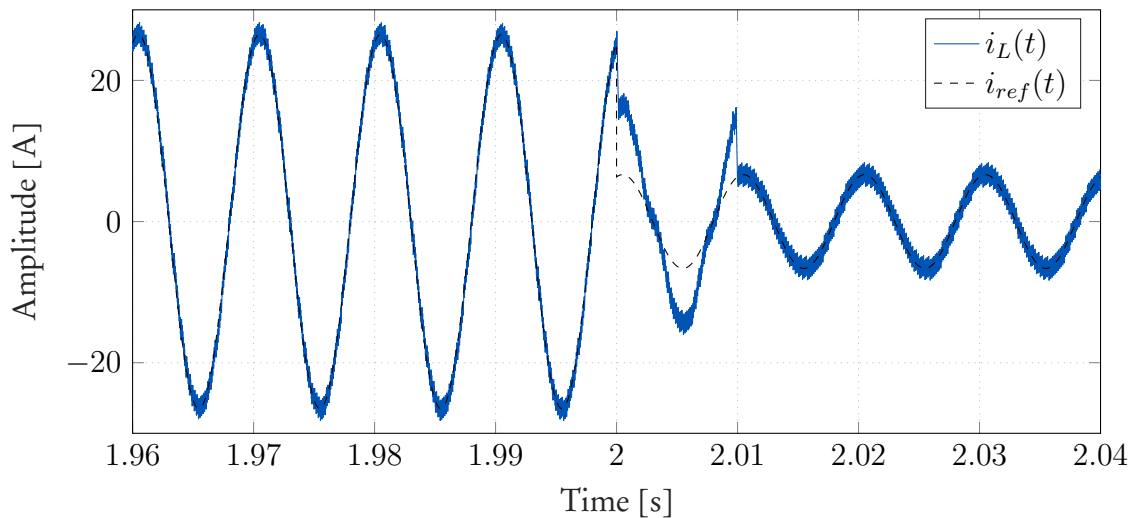


Figure 4.14: Simulation waveform of the inductor current  $i_L(t)$  for a step-down change in the load current at  $t = 2$  s. The reference ripple current  $i_{ref}(t)$  is denoted by the black dashed line.

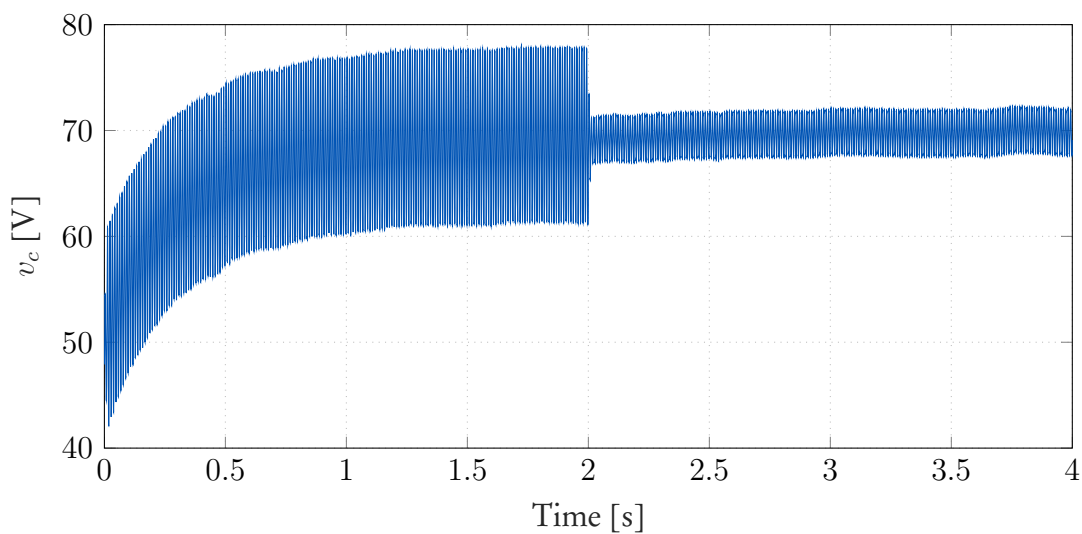


Figure 4.15: Simulation waveform of the active capacitor voltage  $v_c(t)$  for a step-down change in the load current at  $t = 2$  s.

The battery current  $i_b(t)$  resulting from the step-down change is shown in Figure 4.16. An undershoot is observed immediately after introducing the step-down change, and after about 10 ms, another transient occurs at  $t = 2.01$  s. The battery current immediately goes into a steady-state, similar to the transient response of the battery current to a step-up change in the load current.

The amplitude of the second-order harmonic component is measured for both times before and after the step-down change as,  $I_{100\text{Hz}} = 0.70$  A and  $I_{100\text{Hz}} = 0.36$  A, from an initial value of,  $I_{100\text{Hz}} = 26.50$  A and  $I_{100\text{Hz}} = 6.64$  A, respectively.

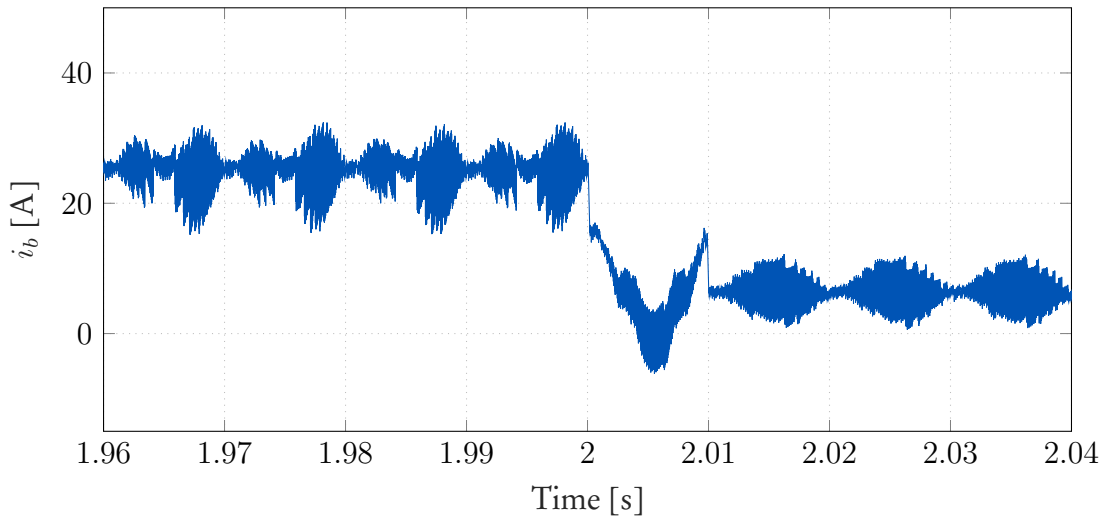


Figure 4.16: Simulation waveform of the current  $i_b(t)$  through the battery for a step-down change in the load current at  $t = 2$  s.

It can be concluded that the controller has a relatively short settling time in the order of 10 ms, for both cases, the step-up and step-down change. A second transient is noticed in the inductor current  $i_L(t)$  and battery current  $i_b(t)$ , for both cases 10 ms after the initial step change is applied to the system, in which the system instantly goes into steady-state operation without any damping oscillations. The control objective of reducing the ripple current passing through the battery is achieved even after a transient has occurred in the system.

## 4.5 Summary

In this chapter, simulations via MATLAB® Simulink were carried out. The simulations were done to verify the effectiveness of the proposed MPC strategy for ripple energy compensation in a single-phase dc-to-ac converter using a boost converter. Both a grid-connected and stand-alone system were used to perform the simulations.

At a switching frequency of approximately 16 kHz for the boost converter, the ripple current passing through the battery was reduced to roughly 2.4% and 2.8% of the nominal battery current for the grid-connected system and stand-alone system, respectively, using a prediction horizon of  $N_p = 10$ . The transient response of the model predictive controller was investigated, and a fast response to step changes was demonstrated. Also, for both systems, the capacitance requirements were significantly reduced by over 95% when using this method of ripple energy compensation.

It was observed that the model predictive controller achieved its primary objectives for both systems. The inductor current accurately tracked the reference ripple current, and the

ripple current flowing through the battery was reduced to much less than the recommended value, that is, 10 % of the nominal battery current. However, the control of the active capacitor voltage was somewhat challenging. The active capacitor voltage was too sluggish to follow the time-varying voltage reference. The anticipated reasons for this behaviour were associated with the contradiction between the reference signals used in the controller and the lack of a better reference for the active capacitor voltage.

The performance benefit of utilizing long-prediction horizons to solve the optimization problem was noticed since, for horizons  $N_p < 3$ , the system is unstable. This is due to the non-minimum phase behaviour that the boost converter exhibits. Even though longer horizons offered significant performance benefits to the system, the weighting factor  $\lambda_u$  had to be tuned through trial-and-error to obtain the desired switching frequency. Moreover, since the operating point changes for a given horizon, the weighting factor was adjusted for each of the horizons considered. Hence, the process became monotonous and highlighted the disadvantage of using long prediction horizons.

Furthermore, only one parameter, the weighting factor  $\lambda_u$ , was tuned during the simulations to attain the desired switching frequency for different prediction horizons. The obtained simulation results were used to evaluate the performance benefits of long horizons. However, the sampling interval  $T_s$ , which determines the length of the prediction horizon  $N_p T_s$ , has profound influence on the performance of the system [31]. To account for various sampling intervals and the scalar weights Monte Carlo simulations were performed in [31]. The Monte Carlo simulations enabled the evaluation of the performance benefits of longer horizons for the different switching frequencies considered to become more evident and clear.

# Chapter 5

## FPGA implementation

### 5.1 Introduction

In this chapter, the implementation of long-horizon direct MPC on an FPGA is discussed. The choice of using an FPGA is mainly due to the computational complexity of the control strategy. First and foremost, the chapter gives a detailed description of the FPGA board used and the integrated design environment supported by the FPGA board. Secondly, the implementation in VHDL is discussed. The basic arithmetic concepts concerning VHDL and the notion of computational delay compensation associated with the practical implementation of predictive controllers is explained. To determine the practical feasibility of the controller design in VHDL, a hardware-in-the-loop (HiL) simulation of the converter has to be performed. Thus, the stand-alone system is used. The implementation of the HiL simulation and the controller, which includes both the model predictive controller and the pulse-width modulator, are discussed in detail. After performing the HiL simulation, the results obtained are compared with the MATLAB® simulation results presented in Chapter 4. This way, the FPGA implementation of the controller is verified. Finally, the chapter concludes with a summary.

### 5.2 The FPGA

#### 5.2.1 A brief description of the FPGA board

The ZedBoard, a low-cost evaluation and development board from Digilent Inc, is used to implement the controller. The board is based on the Xilinx® XC7Z020-1CLG484C Zynq-7020 all programmable system on a chip (AP SoC) architecture [32]. The Zynq-7020 AP SoC integrates both a dual-core ARM® Cortex™-A9 processor and 28 nm Xilinx FPGA in a single device [32]. In this thesis, only the Xilinx FPGA is used to implement the controller.

The ZedBoard has an on-board oscillator that operates at 100 MHz. The oscillator generates a clock signal fed to the FPGA through an input pin. Thus, the clock signal required for the implementation of the controller is generated from the on-board oscillator.

#### 5.2.2 FPGA building blocks

The architecture and physical capabilities of an FPGA should be known in order to appreciate its working principles. Therefore, in this section, some of the prominent attributes of the Xilinx FPGA on the Zynq-7020 AP SoC are discussed.

### Input/output blocks

FPGAs have input/output (I/O) pins. More specifically, the Xilinx FPGA has 200 such pins. The I/O pins offer a wide range of standard voltage levels, that is, from 1.2 V to 3.3 V. The I/O pins are grouped in banks of 50, of which two pins in these banks are classified in pairs, as positive and negative. They can be used as single-ended or differential I/O pairs, and can also be used in reference mode [32].

### Configurable logic blocks

Lookup tables (LUTs), flip-flops and multiplexers lie at the heart of configurable logic blocks (CLBs). The CLBs are used as basic elements in the implementation of a digital system on an FPGA. The Xilinx FPGA has 53 200 LUTs and 106 400 flip-flops. The LUTs can be configured in two different ways: either as one 6-input LUT (64-bit ROMs) with a single output or as two 5-input LUTs (32-bit ROMs) with independent outputs that have common logic inputs or addresses [32].<sup>1</sup> Moreover, the LUTs, flip-flops, and multiplexers are grouped as slices, of which, each slice consists of four LUTs, eight flip-flops, multiplexers, and arithmetic carry logic [32]. Two of such slices form a CLB.

### Block RAM

The Xilinx FPGA has 149 dual-port block RAMs.<sup>2</sup> The ports of each block RAM are completely independent of each other and share nothing but the data stored in them. Each block RAM is capable of storing 36 kB data (total block RAM capacity of 4.9 MB) [32]. Furthermore, large LUTs, shift registers, or buffers can be formed from the block RAMs. Every single block RAM of 36 kB can store data with a 64-bit width; moreover, the blocks can generate, store and utilize an extra eight bits to perform error correction during the read or write process (memory access).

### Digital signal processing slices

Digital signal processing (DSP) slices are blocks that are used to carry out arithmetic and logic operations. There are 220 such DSP slices on the Xilinx FPGA. The signal processing operations utilize many multipliers and adders (also referred to as accumulators). These operations are carried out in the dedicated DSP slices, of which each DSP slice comprises of a  $25 \times 18$  bit two's complement multiplier and a 48-bit accumulator [32]. Both of them have the ability to operate up to 741 MHz.

Since there is a finite number of DSP slices, the management of resources in the implementation of a digital system, particularly the multipliers, is of utmost importance. For example, for the model predictive controller implementation, care should be taken when utilizing the resources in order to achieve long prediction horizons.

---

<sup>1</sup>ROM refers to read-only memory.

<sup>2</sup>RAM refers to random-access memory.

### 5.2.3 FPGA preliminaries

#### Hardware description languages

The two most popular hardware description languages (HDLs) used to implement a digital system on an FPGA are:

1. Verilog HDL
2. Very High Speed Integrated Circuits (VHSIC) HDL (VHDL).

Each of the HDLs has advantages and disadvantages; however, the choice between the two depends on the preference of the code developer; there are no major performance or complexity considerations. Henceforth, throughout this chapter, only VHDL will be considered.

#### The integrated design environment

The Xilinx Vivado design suite, i.e., the integrated design environment (IDE), synthesizes, simulates, and implements the HDL description of a digital system (also referred to as the behavioural description of a program). In this chapter, the behavioural description of the program is given by the VHDL code.

Firstly, to realize a digital system on an FPGA, it has to be synthesized. This means representing the behavioural description of a program (the VHDL code) using the elements of the FPGA given in Section 5.2.2, particularly, the logic blocks. Simply put, the VHDL code is transformed into a physical device, i.e., a netlist. The structure of the code determines how the components of the netlist are interconnected. After synthesizing the digital system, it is prepared for implementation on the target FPGA. This is the second step. During this step, the optimization and minimization tools of the IDE reduce the usage of resources while taking the FPGA's physical properties into account. The final step is programming the FPGA. An interface is set up, between the FPGA and the implemented design of the digital system, by a constraint file [33]. The constraint file consists of the pin information such as the switches, clock, light emitting diodes (LEDs), seven-segment display, and buttons of the ZedBoard. The implemented design of the digital system is then fed to the FPGA as a bitstream.

Due to the delays in the FPGAs logic elements, timing constraints can exist, particularly as the complexity of the program increases and becomes more arithmetically intensive. Thus, timing requirements of the FPGA must be taken into consideration in the implementation procedure. The IDE offers a detailed timing summary which notifies the code developer when timing violations have occurred. The conditions of the timing violations are determined by the longest path (in time), i.e., the critical path.

It is essential to verify and observe the internal signals of the implemented design of the digital system. Thanks to Xilinx, the Vivado design suite offers an integrated logic analyzer (ILA) core. The logic analyzer allows the monitoring of signals on the FPGA. Some of the key features and the benefits of the ILA core include boolean trigger equations, edge transition triggers, user-selectable width, data width, and depth, as well as multiple probe ports with the capability of being combined into a single trigger condition [34]. The ILA core is synchronized with the system clock; thus, the components inside the ILA core adhere to the clock constraints applied to the implemented design of the digital system.



## 5.3 VHDL implementation

Recall that the stand-alone system consists of a single-phase dc-to-ac converter with an  $RL$  load and a boost converter. As already stated in Section 4.2, the single-phase dc-to-ac converter is controlled by PWM with unipolar switching.<sup>3</sup> The boost converter is controlled using long-horizon direct MPC.<sup>4</sup>

In this section, initially, a few arithmetic concepts concerning VHDL are discussed, then followed by the notion of delay compensation. The procedure followed to implement the HiL simulation for both the converters in the stand-alone system, the pulse-width modulator, and the model predictive controller, within the FPGA is given.<sup>5</sup> Finally, the FPGA implementation of the proposed model predictive controller is verified by comparing the HiL simulation results with the MATLAB® simulation results to ensure ripple energy compensation is achieved in a practical system.

### 5.3.1 Arithmetic in VHDL

#### Fixed-point numbers

A real value can either be just an integer or an integer with a fractional part. In VHDL, there are two ways to represent an integer with a fractional part. These are fixed-point and floating-point representations.

For the floating-point representation, the integer and fractional parts of a value do not have a fixed number of bits assigned to them; rather, the assigned number of bits differs depending on its significant bits [33]. For this reason, a much larger dynamic range of values (largest and smallest values) can be represented in this form. Moreover, the implementation of the floating-point representation is complex and requires a great deal of computing power.

On the other hand, fixed-point representation has a fixed number of integer and fractional bits, as the name suggests. Thus, a restriction exists regarding the range of numbers that can be represented in this form and their resolution. However, for the implementation of the controller on the FPGA, the fixed-point representation of real values is more than sufficient. It has the advantage of being easy to implement and using less hardware; hence, it performs computations faster than floating-point arithmetic. Although, caution must be taken with fixed-point numbers when rounding off as quantization errors may occur. Furthermore, a trade-off exists between resource usage and accuracy, which results from the accuracy of a value being determined by the number of bits assigned for the fractional part. Hence, a compromise is made between the use of resources and accuracy.

#### Matrices

Matrices are extensively used in the implementation of the HiL simulation and, especially, the model predictive controller. Unfortunately, VHDL does not natively support the use of matrices. Therefore, to overcome this shortcoming, arrays are used to implement matrices. One way of implementing a matrix using arrays is through a two-dimensional array (2D array); an alternative way is through a one-dimensional array in a one-dimensional array (1D array in a 1D array).

<sup>3</sup>Refer to Section 2.1.3 for information on PWM with unipolar switching.

<sup>4</sup>Refer to Chapter 3 for information on the control of boost converters using long-horizon direct MPC.

<sup>5</sup>Note that throughout this chapter when only the term “controller” is used it refers to both the pulse-width modulator and the model predictive controller.

In the 2D array case, elements of the array are basically stored in the same way matrix elements would. While, in the 1D array in a 1D array case, elements in each row of the matrix are stored as an array, as illustrated in the following example,

$$\mathbf{P} = \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix} \Leftrightarrow \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix} .$$

To access a specific element in either of the two matrix implementations, the row is referred to first, followed by the column in which the element resides. The slight difference only comes in the approaches used to access the element based on its index (location) in the array. For example, to access the element in the second row and third column of matrix  $\mathbf{P}$ , i.e., element  $h$ . With the 2D array implementation, the array index will be  $P_{(2,3)}$ , and for the 1D array in a 1D array implementation the array index will be  $P_{(2)(3)}$ . It is important to choose one array implementation of the matrix and stick with it when coding. Thus, only the 1D array in a 1D array indexing will be considered hereafter.

### 5.3.2 Computation delay compensation

Thus far, the idealized case of predictive control has been assumed. The time delay between sampling instants and the application of an optimal switch position at the corresponding sampling instants has been ignored. This case is illustrated in Figure 5.1.

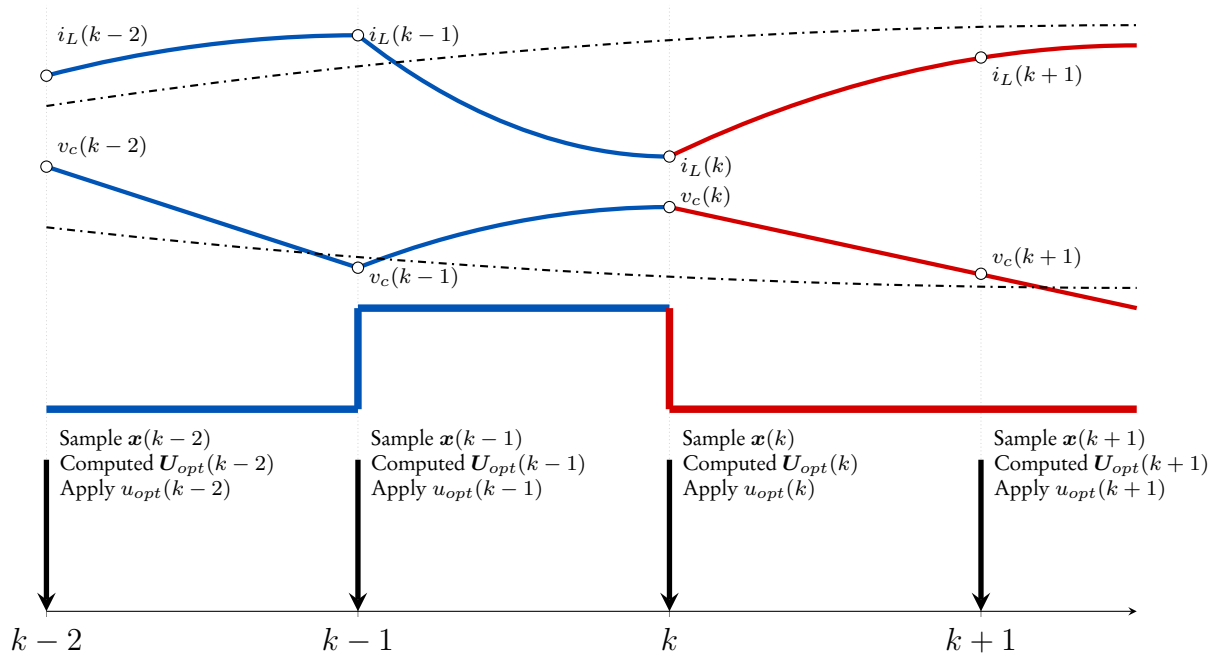


Figure 5.1: Example of the ideal scenario with zero computation delay. The top dash-dotted black line and bottom dash-dotted black line represent the reference ripple current and active capacitor voltage reference, respectively. Before the sampling instant  $k$ , the inductor current, the active capacitor voltage and the switching positions are represented by the blue lines (the historical or rather, the actual data). After the sampling instant  $k$ , the optimal switching positions and their corresponding inductor current and active capacitor voltage trajectories are represented by red lines (predictions).

The inductor current and active capacitor voltage of the boost converter are sampled at time-step  $k$ . Once the measured samples are taken and encompassed in a state vector  $\mathbf{x}(k)$ , the optimal switching sequence  $\mathbf{U}_{opt}(k)$  is instantaneously calculated, and the corresponding optimal switch position  $u_{opt}(k)$  is applied to the system at the current sampling instant. The same procedure is repeated for the successive sampling instants in the prediction horizon.

In a simulation, the ideal discrete-time setup is possible. However, in a practical setup, a time delay exists between a sampling instant and the application of the optimal switch position to the system. Some of the sources of delay given in [11] include:

- *Measurement delay.* The acquisition and conversion of the measured signals by the analog-to-digital converter (ADC) introduces a delay.
- *Computation delay.* The computation of the optimal switching sequence takes a certain amount of time to be executed entirely. Also, often the available FPGA resources are shared with other processes. As a result, in this case, a reduced amount of computation power will be available for the implementation of the model predictive controller, which in turn increases the computation delay.
- *Communication delay.* The transmission of the digital measurements from the ADC to the FPGA incurs a communication delay. Furthermore, when an optimal switch position is obtained, it is sent back to the system, introducing a communication delay.

The computation delay depends on how fast the non-recursive MPC algorithm, Algorithm 5, which incorporates the branch-and-bound technique, traverses the search tree. As a result, the computation delay time  $t_c$  is *variable*. It should be noted that switching between the sampling instants is not allowed, as emphasized in Figure 5.2. Once switching transitions occur between sampling instants, the prediction model is not valid anymore.

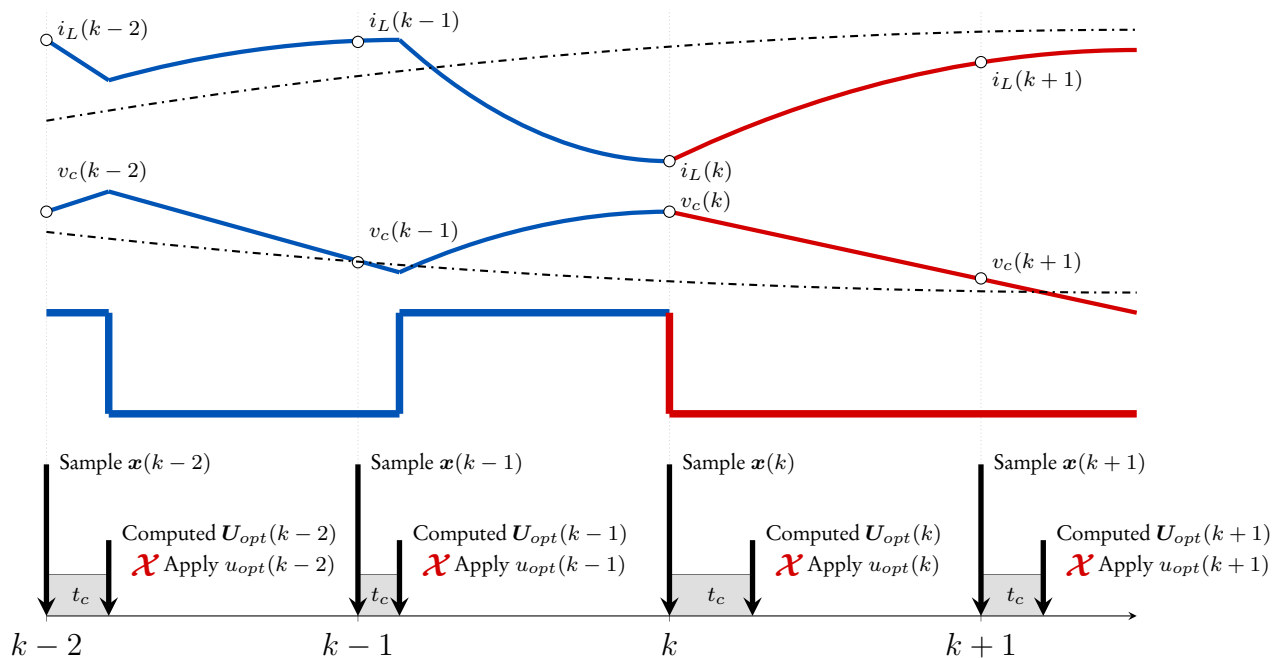


Figure 5.2: Example of the practical scenario with a computation delay time  $t_c$ . The red  $\mathcal{X}$  symbol indicates that after computing the switching sequence for a particular sampling instant, the optimal switch position for that sequence cannot be immediately applied to the system between two sampling instants.

Knowing that the computation delay time  $t_c$  is variable, a fixed time delay  $T_d$  between the sampling of measurements and the application of the optimal switch position can be assumed. The fixed time delay,  $T_d = T_s$ , is equivalent to one sampling interval. This fixed time delay implies that, when the optimal switching sequence is calculated after a particular computation time  $t_c$ , using samples at say the time-step  $k$ , the optimal switch position  $u_{opt}(k)$  will only be applied at time-step  $k + 1$ . In other words, the optimization problem is not solved to find  $\mathbf{U}_{opt}(k)$  at the *current* sampling instant, instead, it is solved to find  $\mathbf{U}_{opt}(k + 1)$ , the optimal switching sequence for the *next* sampling instant. To state that, the notation  $\mathbf{U}_{opt}(k + 1|k)$  is used. At the current sampling instant, the optimal switch position  $u_{opt}(k|k - 1)$  is applied, which was calculated at the *previous* sampling instant.

For the control of the boost converter, the predicted state variables at the next sampling instant are, therefore, given by<sup>6</sup>

$$\mathbf{x}(k + 1|k) = \begin{cases} \mathbf{A}_0\mathbf{x}(k) + \mathbf{B}_0u_{opt}(k|k - 1) \\ \mathbf{A}_1\mathbf{x}(k) + \mathbf{B}_1u_{opt}(k|k - 1). \end{cases} \quad (5.1)$$

The optimization problem is solved using the measurements at the current sampling instant, for the next sampling instant. Now, the optimization problem is stated as

$$\begin{aligned} \mathbf{U}_{opt}(k + 1|k) = \arg \min_{\mathbf{U}(k+1)} J \\ \text{subject to (5.1),} \\ \mathbf{U}(k + 1) \in \{0, 1\}^{N_p}. \end{aligned} \quad (5.2)$$

Upon solving the optimization problem in (5.2) within the sampling interval, the optimal switching sequence  $\mathbf{U}_{opt}(k + 1|k)$  is obtained. Only when the next sampling instant is reached, the optimal switch position  $u_{opt}(k + 1|k)$  is applied. The procedure is shown in Figure 5.3.

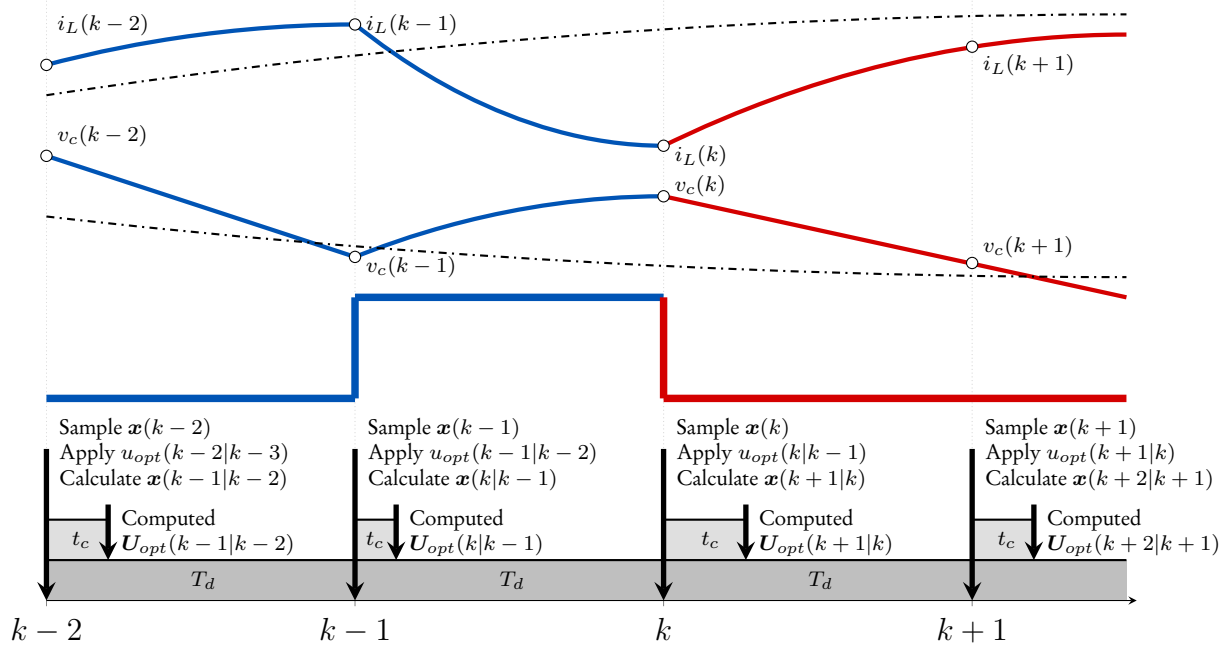


Figure 5.3: Illustration of computation delay compensation.

<sup>6</sup>The notation is explained: At time-step  $k + m$ , the value of the state vector is  $\mathbf{x}(k + m|k + m - 1)$ , as calculated at time step  $k + m - 1$ . The optimal switch position that was applied at time-step  $k + m - 1$  is  $u_{opt}(k + m - 1|k + m - 2)$  obtained from measurements at time-step  $k + m - 2$ .

### 5.3.3 Hardware-in-the-loop simulation

The so-called HiL simulation can test the practical feasibility of the controller design. Without building the actual physical hardware of the practical system for the experimental setup, a HiL simulation can be implemented within an FPGA and, thus, assume the role of the practical system. HiL simulation has been widely used over the years in the industry, especially for complex systems, due to its advantages.

For complex systems, if the controller malfunctions, possibly due to bugs in the code, catastrophic damage can be done to the physical hardware. Therefore, a HiL simulation can validate the controller before running it on the physical hardware. When there are software enhancements made to the controller, a HiL simulation can be used to re-run functionality tests to ensure that the controller works as desired. This is referred to as *regression testing*. For these reasons, it can be seen that a HiL simulation has the advantage of reducing the likelihood of encountering unforeseen damage to the physical hardware and potential safety hazards.

In many development projects carried out, the development time is of great essence in completing the entire project. Usually, the implementation of the controller is complete before the procurement of the hardware components. In that case, a HiL simulation can test the controller in the meantime, thus, reducing the overall development time.

Another advantage worth mentioning is that more information on the signals and the general working of the practical system can be acquired from a HiL simulation. Obtaining this information on the physical system would require measurement equipment that might be expensive or difficult to come by. Hence, the cost of testing is reduced.

#### Implementation of the HiL simulation

For convenience to the reader, the topology of the stand-alone system is shown in Figure 5.4.

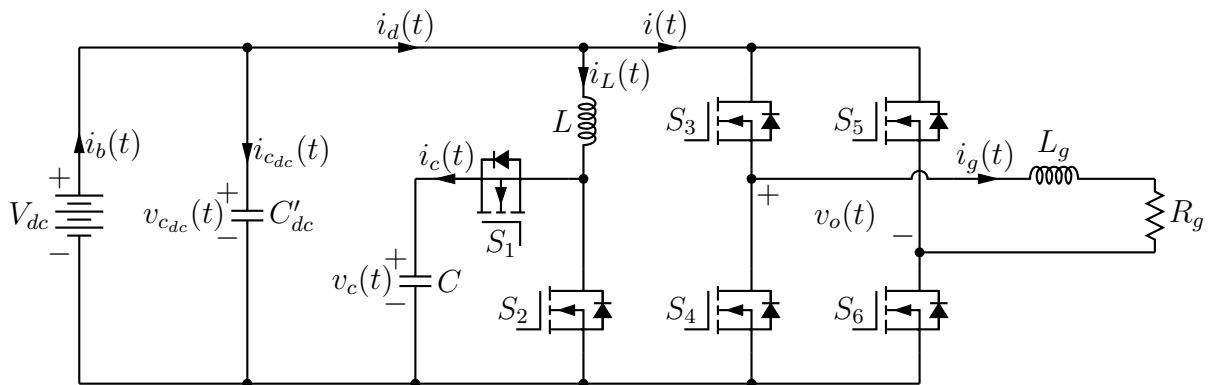


Figure 5.4: Topology of the stand-alone system.

In VHDL, a process is created that acts as an “internal” HiL. The process simulates the current and voltage signals of the stand-alone system. The signals produced as a result of controlling the single-phase dc-to-ac converter are stored as variables that are local to the HiL process. Whereas, the signals that result from the control of the boost converter, i.e., the state variables, are stored in a global array, for a reason that will soon become apparent.

Note that, if not stated otherwise, all of the processes in this chapter are synchronized with the system clock that runs at a frequency  $f_{clk}$ . From the system clock, sub-clocks with the desired frequency can be generated inside a process using counters. Thus, for the HiL process,

a sub-clock that has a sampling frequency  $f_{hil}$ , is generated and used inside the process. At every sampling instant, gating signals are sent to the HiL process from the model predictive controller and pulse-width modulator.

Recall that the single-phase dc-to-ac converter has an *open-loop* controller. On the other hand, the model predictive controller for the boost converter has a *closed-loop* controller. With that being said, when the model predictive controller computes the optimal switch position  $u_{opt}(k)$  at the current sampling instant, the optimal switch position is sent to the HiL process at the next sampling instant. Thereafter, the HiL process updates the state variables upon receiving the optimal switch position  $u_{opt}(k+1|k)$ . The updated state variables are fed to the model predictive controller after the delay compensation, as explained in Section 5.3.2. Thus, the feedback loop is closed.

It can be noticed that the state variables of the boost converter are used in the HiL process and to implement the model predictive controller, hence, the reason for storing them in a global array.

### Discretized model of the stand-alone system

The HiL simulation requires a discretized model of the stand-alone system in order to simulate the current and voltage signals. This means the discretized model of both the single-phase dc-to-ac converter and the boost converter are required.

In Section 3.2, the discretization of the boost converter is explained. The only difference is the sampling interval for the discretization of the continuous-time state-space representation in (3.1). Instead of using the sampling interval  $T_s$  as used for discrete-time state-space representation for the model predictive controller, the sampling interval for the HiL process,  $T_{hil} = \frac{1}{f_{hil}}$ , is used. Hence, the discrete-time state-space representation is given by<sup>7</sup>

$$\mathbf{x}(k+1|k) = \begin{cases} \mathbf{A}_{h0}\mathbf{x}(k) + \mathbf{B}_{h0}u_{opt}(k+1|k) \\ \mathbf{A}_{h1}\mathbf{x}(k) + \mathbf{B}_{h1}u_{opt}(k+1|k), \end{cases} \quad (5.3)$$

where

$$\mathbf{A}_{h0} = \mathbf{I}_2 \quad (5.4a)$$

$$\mathbf{A}_{h1} = e^{\mathbf{F}_1 T_{hil}} \quad (5.4b)$$

$$\mathbf{B}_{h0} = T_{hil} \mathbf{G} \quad (5.4c)$$

$$\mathbf{B}_{h1} = -\mathbf{F}_1^{-1}(\mathbf{I}_2 - \mathbf{A}_{h1})\mathbf{G}. \quad (5.4d)$$

The first step in the discretization of the single-phase dc-to-ac converter is to consider its output stage. The output voltage  $v_o(t)$ , load current  $i_g(t)$ , and an  $RL$  load, denoted by  $R_g$  and  $L_g$ , is extracted from Figure 5.4, and is shown in Figure 5.5.

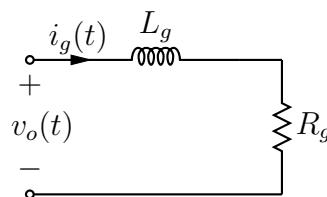


Figure 5.5: Output stage of the single-phase dc-to-ac converter.

<sup>7</sup>For the state matrices  $\mathbf{F}_0$ ,  $\mathbf{F}_1$  and  $\mathbf{G}$  of the boost converter, the reader is referred to Section 3.2.

The load current is described by the differential equation

$$\dot{i}_g(t) = F_g i_g(t) + G_g u_{pwm}(t), \quad (5.5)$$

where

$$F_g = -\frac{R_g}{L_g} \quad (5.6a)$$

$$G_g = \frac{V_{dc}}{L_g}, \quad (5.6b)$$

and  $u_{pwm} \in \{-1, 0, 1\}$  denotes the switching function of the converter that results from PWM with unipolar switching. The current is discretized using exact discretization [11]. The obtained discrete-time state-space representation

$$i_g(k+1) = A_g i_g(k) + B_g u_{pwm}(k), \quad (5.7)$$

where

$$A_g = e^{F_g T_{hil}} \quad (5.8a)$$

$$B_g = -F_g^{-1}(1 - A_g)G_g. \quad (5.8b)$$

with  $e^{F_g T_{hil}}$  being the exponential of  $F_g T_{hil}$ .

If a lossless system is assumed, then, the discretized rectified current is described by

$$i(k) = i_g(k)u_{pwm}(k). \quad (5.9)$$

At the input stage of the single-phase dc-to-ac converter, in order to simulate the battery current  $i_b(t)$ , a resistor  $R_{dc}$  with an extremely low resistance is placed between the dc supply voltage  $V_{dc}$ , and the dc-link capacitor  $C'_{dc}$ . The input stage is extracted from Figure 5.4, and is shown in Figure 5.6.

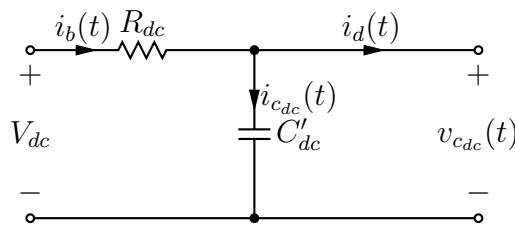


Figure 5.6: Input stage of the single-phase dc-ac converter.

The differential equation that describes the voltage across the dc-link capacitor is

$$\dot{v}_{cdc}(t) = F_v v_{cdc}(t) + G_{v1} i_d(t) + G_{v2} V_{dc}, \quad (5.10)$$

where

$$F_v = -\frac{1}{R_{dc}C'_{dc}} \quad (5.11a)$$

$$G_{v1} = -\frac{1}{C'_{dc}} \quad (5.11b)$$

$$G_{v2} = \frac{1}{R_{dc}C'_{dc}}. \quad (5.11c)$$

The current  $i_d(t)$  shown in (5.10) is defined by

$$i_d(t) = i(t) + i_L(t). \quad (5.12)$$

By using exact discretization, the discretized voltage across the dc-link capacitor

$$\dot{v}_{c_{dc}}(k+1) = A_v v_{c_{dc}}(k) + B_{v1} i_d(k) + B_{v2} V_{dc}, \quad (5.13)$$

where

$$A_v = e^{F_v T_{hil}} \quad (5.14a)$$

$$B_{v1} = -F_v^{-1}(1 - A_v)G_{v1} \quad (5.14b)$$

$$B_{v2} = -F_v^{-1}(1 - A_v)G_{v2}. \quad (5.14c)$$

Thus, the discretized battery current is given by

$$i_b(k) = \frac{V_{dc} - v_{c_{dc}}(k)}{R_{dc}}. \quad (5.15)$$

### Formulation of a digital filter

As already mentioned in Section 3.3.3, the correct amplitude and phase angle of the second-order harmonic component is required for ripple energy compensation to be possible. The rectified current  $i(t)$  is passed through a second-order band-pass filter and the ripple current  $i_r(t)$  obtained during steady-state operation is set to  $180^\circ$  out of phase with second-order harmonic component. This way the reference ripple current  $i_{ref}(t)$  is calculated.

The transfer function of the band-pass filter in (3.10) is given as

$$H_f(s) = \frac{H_0 \frac{\omega_f}{Q_f} s}{s^2 + \frac{\omega_f}{Q_f} s + \omega_f^2}.$$

The filter can be described by the continuous-time state-space representation

$$\dot{\mathbf{x}}_f(t) = \mathbf{F}_f \mathbf{x}_f(t) + \mathbf{G}_f i(t), \quad (5.16)$$

where the state vector includes the ripple current and its derivative,

$$\mathbf{x}_f(t) = [i_r(t) \quad \dot{i}_r(t)]^T. \quad (5.17)$$

The state and input matrices are defined by

$$\mathbf{F}_f = \begin{bmatrix} 0 & 0 \\ -\omega_f^2 & -\frac{\omega_f}{Q_f} \end{bmatrix} \quad (5.18a)$$

$$\mathbf{G}_f = \begin{bmatrix} H_0 \frac{\omega_f}{Q_f} \\ -H_0 \left(\frac{\omega_f}{Q_f}\right)^2 \end{bmatrix}. \quad (5.18b)$$

The interested reader is referred to [35], for a detailed explanation of the continuous-time state-space representation of a transfer function. The discrete-time state-space representation of the filter is obtained by exact discretization as

$$\mathbf{x}_f(k+1) = \mathbf{A}_f \mathbf{x}_f(k) + \mathbf{B}_f i(k), \quad (5.19)$$

where

$$\mathbf{A}_f = e^{\mathbf{F}_f T_{hil}} \quad (5.20a)$$

$$\mathbf{B}_f = -\mathbf{F}_f^{-1}(\mathbf{I}_2 - \mathbf{A}_f)\mathbf{G}_f. \quad (5.20b)$$



### 5.3.4 Controller implementation

#### Pulse-width modulator

PWM with unipolar switching is employed as the switching strategy for the semiconductor switches of the single-phase dc-to-ac converter. Thus, two sinusoidal reference signals that are out of phase with each other by  $180^\circ$ , and a triangular carrier signal are required to implement the pulse-width modulator within the FPGA.

The sinusoidal reference signals are pre-calculated offline for one fundamental period, i.e.,  $T_1 = \frac{1}{f_1}$ , at a chosen discrete sampling interval defined by  $T_{ref}$ .<sup>8</sup> The discretized sinusoidal reference signals are loaded onto the ROM of the FPGA for online use.<sup>9</sup> It is important to note that the narrower the chosen sampling interval  $T_{ref}$  is, the more samples are produced. Hence, the more accurate the sinusoidal signals are. Nonetheless, care should be taken when choosing the sampling interval that best represents the sinusoidal waveforms since there are limited resources on the FPGA, i.e., the Block RAM primitives. To access the sinusoidal reference signals from the ROM, a process is created that obtains the sinusoidal reference signals by the address of their samples in the ROM using counters. A sub-clock used in the process has a clock cycle with a period  $T_{ref}$ .

The triangular carrier signal is generated in another process using a counter. The counter counts up from a minimum value to a maximum value, and subsequently, counts down from the maximum value to the minimum value, repeatedly. Let  $cnt_{tri}$ ,  $min_{val}$  and  $max_{val}$  denote the counter, minimum value and maximum value, respectively. A flag – “direction” – is set, it decides whether  $cnt_{tri}$  should count up or down when it reaches either  $max_{val}$  or  $min_{val}$ . The procedure for generating the triangular signal is illustrated as an automation, as shown in Figure 5.7.

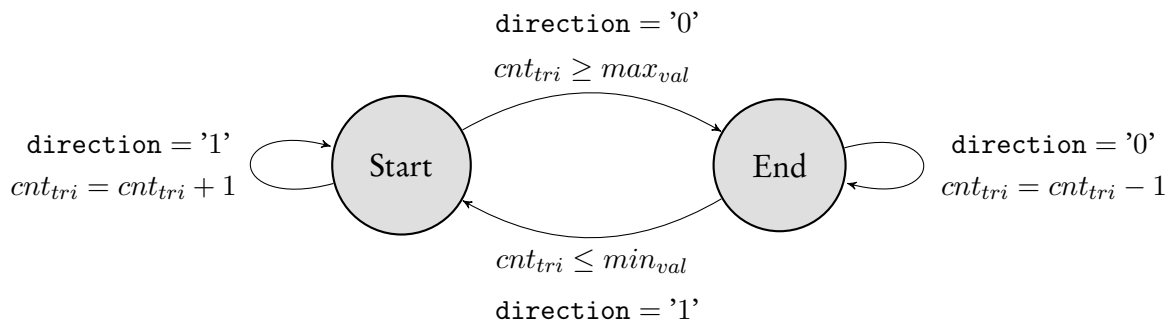


Figure 5.7: Illustration of the triangular signal generation procedure as an automation.

To obtain the desired frequency  $f_c$  of the triangular carrier signal, the counter  $cnt_{tri}$  must count  $N_{cnt}$  times in every single period,  $T_c = \frac{1}{f_c}$ , of the triangular signal, where

$$N_{cnt} = \frac{f_{clk}}{f_c}. \quad (5.21)$$

After every clock cycle, with a period  $T_{clk} = \frac{1}{f_{clk}}$ , the counter  $cnt_{tri}$  increments by one unit from  $min_{val} = -\frac{N_{cnt}}{4}$  until it reaches  $max_{val} = \frac{N_{cnt}}{4}$ , then it starts to decrement again by one unit for  $\frac{N_{cnt}}{2}$  times. This is demonstrated in Figure 5.8.

<sup>8</sup>Note that offline means that the calculations are not done on the FPGA, there are done in MATLAB®, this will be mentioned later.

<sup>9</sup>Note that online means that the calculations are done on the FPGA.

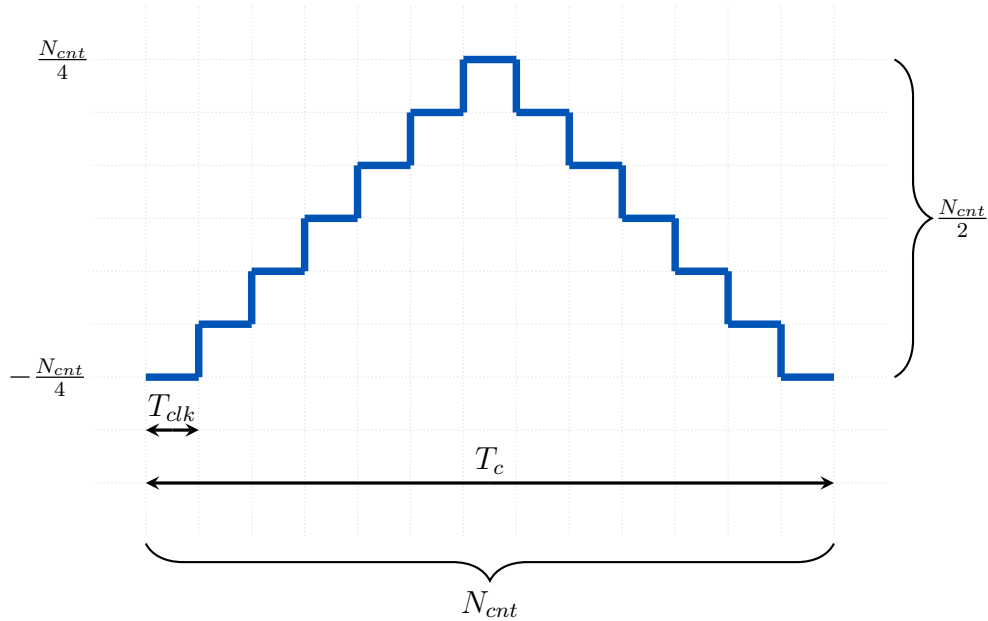


Figure 5.8: Demonstration of how the triangular carrier signal is created on an FPGA.

Note that the modulation index  $m_a$  of the sinusoidal reference signals is scaled by  $\frac{N_{cnt}}{4}$  during the offline calculation. The triangular carrier signal is compared with the sinusoidal reference signals to generate pulse-width modulated signals for each of the four semiconductor switches in the converter. By using the switching positions of the generated pulse-width modulated signals, the switching function  $u_{pwm}(k)$ , which is used (5.7) and (5.9), for the HiL simulation, is calculated in another process.

The reader is referred to Section 2.1.3 for a detailed explanation of the generation of a switching function using the switch positions of pulse-width modulated signals. Specifically, when the unipolar switching strategy is employed.

### Model predictive controller

To implement the model predictive controller, the vector  $\mathbf{x}_{ref}(k)$  encompassing the reference signals used for the optimal control of the boost converter state variables  $\mathbf{x}(k)$ , is required. By using the discrete-time state-space model of the second-order band-pass filter in (5.19), the amplitude and phase angle of the ripple current  $i_r(t)$  is obtained from the state vector  $\mathbf{x}_f(k)$  in the HiL simulation when it reaches steady-state operation. From the parameters obtained, the reference ripple current  $i_{ref}(k)$  is calculated offline at discrete intervals of  $T_s$  and stored in an array. The active capacitor voltage reference  $v_{ref}(k)$  is calculated offline at discrete intervals of  $T_s$  using (2.28) and stored in an array. Note that only one cycle, i.e.,  $\frac{T_r}{T_s}$  samples, of the reference signals is stored in the arrays.<sup>10</sup> These samples correspond to one period of the second-order harmonic component.

In a process, two pointers that conform to the move blocking strategy are created.<sup>11</sup> The purpose of the pointers is to load the samples of the reference ripple current and the active capacitor voltage reference into a global array (1D array in a 1D array) of dimension  $N_p$ .

<sup>10</sup>Recall that  $T_r$  denotes the period of the second-order harmonic component.  $T_r = \frac{1}{f_r}$ , where  $f_r$  is the frequency of the second-order harmonic component, which is twice the fundamental frequency  $f_1$ .  $T_s$  is the sampling interval of the model predictive controller.

<sup>11</sup>The prediction horizon  $N_p$  is split into two parts. The first part with  $N_1$  steps is sampled finely, and the second part of the horizon with  $N_2$  steps is sampled more coarsely.

Recall that  $N_p$  is the length of the prediction horizon. The reference signals are encompassed into a global array  $\mathbf{x}_{ref}(k)$ . Figure 5.9 illustrates the placement of the samples of the reference signals in the global array. The first  $N_1$  elements of the global array are sampled at discrete intervals of  $T_s$  and the second  $N_2$  elements are sampled at  $T'_s = n_s T_s$ , where  $n_s \in \mathbb{N}^+$  (see Section 2.5).

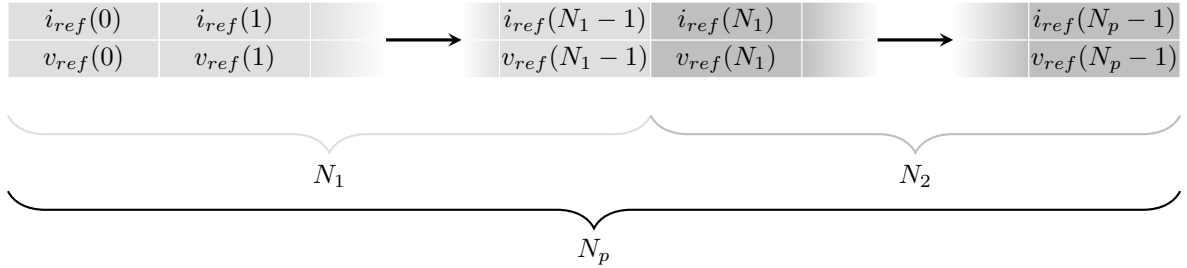


Figure 5.9: Illustration of how the reference signals are placed in an array  $\mathbf{x}_{ref}(k)$  when the prediction horizon of  $N_p$  is split into two parts,  $N_1$  and  $N_2$  time-steps, in accordance with the move blocking strategy.

By using Algorithm 4 in Section 3.3.2, the initial upper bound cost  $J_{ini}$  is calculated. It is used to warm-start the optimization process in the realization of the branch-and-bound technique (see Section 2.4). This calculation is done in a process. The process receives the state variables  $\mathbf{x}(k)$  of the boost converter, and the reference signals  $\mathbf{x}_{ref}(k)$ .

To achieve long-horizons, the initial upper bound cost has to be calculated efficiently on the FPGA. From Algorithm 4,

$$J_{ini} = \sum_{l=k}^{k+N_p-1} \|\mathbf{x}_{ref}(l+1) - \mathbf{x}(l+1)\|_Q^2 + \lambda_u \|U_{ini}(l-1) - U_{ini}(l)\|_2^2, \quad (5.22)$$

with

$$l = k, \dots, k + N_p - 1.$$

If all the entries of  $J_{ini}$  are calculated in one clock cycle in the process, a significant amount of resources on the FPGA will be used. Actually, it is quite improbable that the calculations for all entries will be completed by doing so, due to the intensive arithmetic involved. To solve this shortcoming, only one entry of  $J_{ini}$  is calculated at every clock cycle. Considering that the  $j$ 'th entry of every entry of  $J_{ini}$  is in the exact same form, only the coefficients are changed at every clock cycle. This way the same multipliers are recycled, and used to calculate all the entries of  $J_{ini}$  at every clock cycle. As a result, the resource usage is reduced significantly.

By manipulating (5.22), the  $j$ 'th entry of  $J_{ini}$  is given by

$$\begin{aligned} J_{ini_j} = & J_{ini_j} + (x_{ref}(k+1)_{(0)(j)} - x(k+1)_{(0)(j+1)})Q_{(0)(0)}(x_{ref}(k+1)_{(0)(j)} - x(k+1)_{(0)(j+1)}) \\ & + (x_{ref}(k+1)_{(1)(j)} - x(k+1)_{(1)(j+1)})Q_{(1)(0)}(x_{ref}(k+1)_{(0)(j)} - x(k+1)_{(0)(j+1)}) \\ & + (x_{ref}(k+1)_{(0)(j)} - x(k+1)_{(0)(j+1)})Q_{(0)(1)}(x_{ref}(k+1)_{(1)(j)} - x(k+1)_{(1)(j+1)}) \\ & + (x_{ref}(k+1)_{(1)(j)} - x(k+1)_{(1)(j+1)})Q_{(1)(1)}(x_{ref}(k+1)_{(1)(j)} - x(k+1)_{(1)(j+1)}) \\ & + \lambda_u |U_{ini}(k-1)_j - U_{ini}(k)_j|, \end{aligned} \quad (5.23)$$

where  $\xi_j$  indicates the  $j$ 'th element in a 1D array (i.e., a vector) and  $\xi_{(i)(j)}$  indicates the  $(i')$  $(j')$ th entry of a 1D array in a 1D array (i.e, a matrix).

It is important to note that, in (5.23), for the first entry, the term that penalizes the switching effort is given by

$$\lambda_u |u(k-1) - U_{ini}(k)_j|$$

instead of

$$\lambda_u |U_{ini}(k-1)_j - U_{ini}(k)_j|.$$

Once  $J_{ini}$  is calculated, it is sent to another process where the optimization problem is solved. The process receives  $\mathbf{x}(k)$  and  $\mathbf{x}_{ref}(k)$ . In the process, the non-recursive MPC algorithm that employs the branch-and-bound technique and the move blocking strategy is implemented. The code implementation of the non-recursive MPC algorithm is shown in Algorithm 5 (see Section 3.3.2). From this process, the optimal switching sequence  $\mathbf{U}_{opt}(k)$ , is obtained, and the first switching command  $u_{opt}(k)$  is sent to the HiL simulation at the next sampling instant.

### 5.3.5 Computational burden

#### Offline calculations

As stated in Section 5.3.4, the sinusoidal reference signals used by the pulse-width modulator, and the reference signals used by the model predictive controller, are pre-calculated in advance. They are stored on the FPGA in the ROM and in arrays, respectively. Likewise, the state-space matrices  $\mathbf{A}_0$ ,  $\mathbf{A}_1$ ,  $\mathbf{B}_0$  and  $\mathbf{B}_1$  used in the processes that constitute the model predictive controller are pre-calculated offline and stored in arrays. The matrices  $\mathbf{A}_f$ ,  $\mathbf{B}_f$ ,  $\mathbf{A}_{hil0}$ ,  $\mathbf{A}_{hil1}$ ,  $\mathbf{B}_{hil0}$  and  $\mathbf{B}_{hil1}$ ; the variables  $A_g$ ,  $B_g$ ,  $A_v$ ,  $B_{v1}$ , and  $B_{v2}$ , used in the HiL simulation are also pre-calculated and stored in arrays. The calculations are done offline via MATLAB®. All the calculated values used either just as variables or stored in arrays are converted to fixed-point representation.

Supposing that the calculations mentioned above are done online, a significant amount of resources would be used up. Thus, achieving long-horizons would become impracticable. Recall that the Xilinx FPGA only has 220 DSP slices. Therefore, doing the calculations offline means saving the available resources on the FPGA.

#### Online calculations

Within a sampling period  $T_s$ , there is a permissible number of cycles to perform the online calculations of the model predictive controller. Since the processes that constitute the model predictive controller run at the system clock frequency  $f_{clk}$ , the permitted number of clock cycles within a sampling period is given by  $T_s f_{clk}$ . To ensure optimality, the non-recursive MPC algorithm must finalize its calculations within the allocated clock cycles.

Provided that the non-recursive MPC algorithm does not finalize its calculations within the sampling period, the best solution found during the search process can be applied to the system.

The number of clock cycles allocated to each of the processes that constitutes the model predictive controller, to perform online calculations, is monitored by a finite-state machine (FSM). The FSM is implemented in a process. Inside the process, a counter  $cnt_{cycles}$  performs the state transitions when the clock cycles allocated to a particular process have passed. In each state of the FSM, a flag that triggers a particular process to perform its calculations is set. The FSM and the flags set are shown in Figure 5.10.

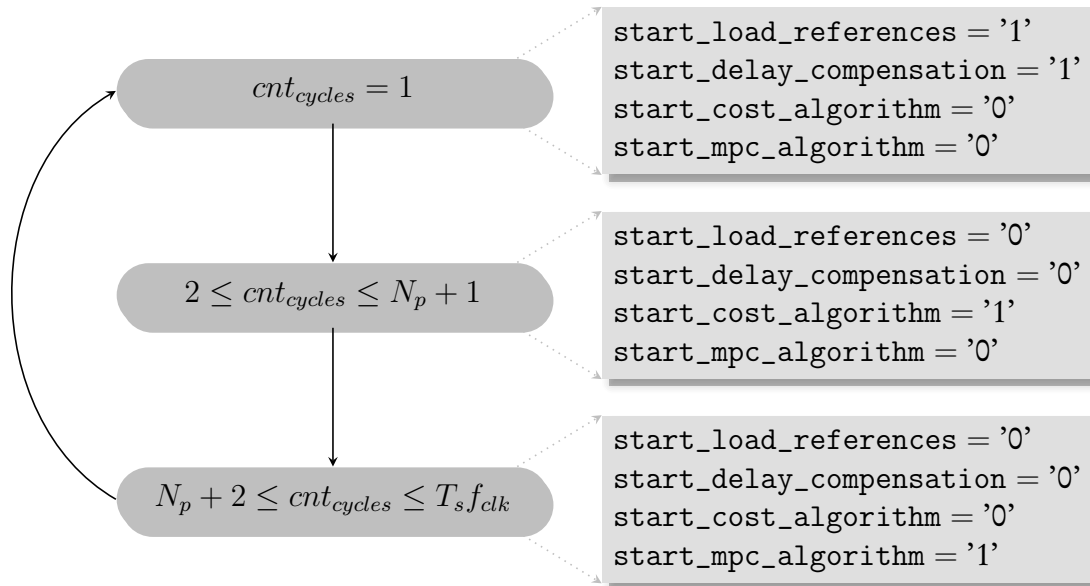


Figure 5.10: A finite-state machine (FSM) that monitors the number of clock cycles allocated to the processes that constitute the model predictive controller.

Below is a summary of the processes that constitute the model predictive controller and the flags used to trigger them:

- $\mathcal{P}_1$ : Process for loading the reference signals into a global array  $\mathbf{x}_{ref}(k)$ .
  - flag:  $start\_load\_references$
- $\mathcal{P}_2$ : Process for implementing delay compensation.
  - flag:  $start\_delay\_compensation$
- $\mathcal{P}_3$ : Process for calculating the initial upper bound cost  $J_{ini}$ .
  - flag:  $start\_cost\_algorithm$
- $\mathcal{P}_4$ : Process for implementing the non-recursive MPC algorithm.
  - flag:  $start\_mpc\_algorithm$

One clock cycle is allocated to  $\mathcal{P}_1$  and  $\mathcal{P}_2$ . The initial upper bound cost  $J_{ini}$  calculation requires  $N_p$  clock cycles. Thus,  $\mathcal{P}_3$  is allocated  $N_p$  clock cycles. Before the non-recursive MPC algorithm starts to solve the optimization problem, already  $N_p + 1$  out of the  $T_s f_{clk}$  clock cycles permitted within a sampling period will have been used. As a result,  $\mathcal{P}_4$  has to finalize its calculations within the remaining clock cycles. The number of cycles allocated to the respective processes are shown in Figure 5.10.

Another counter  $cnt_{samples}$ , which is used in the same process as the FSM and in the process  $\mathcal{P}_1$ , is used to update the reference signals. When the permitted clock cycles allocated to perform the online calculations have passed, and the optimal switching sequence is found,  $cnt_{samples}$  increments, and the reference signals in the global array are updated. The counter is reset after all the samples in one period of the second-order harmonic component have been updated, that is, when  $cnt_{samples} = \frac{T_r}{T_s}$ .

The processes that constitute the pulse-width modulator and the HiL simulation process are always running, in parallel with the model predictive controller.

### 5.3.6 VHDL verification

First and foremost, the stand-alone system and the controller are implemented using MATLAB® scripts. For the stand-alone system, the discretized model presented in Section 5.3.3 is used for the implementation, with the same parameters in given Table 4.3. To verify that the model is correctly implemented, the simulation waveforms produced are compared with simulations done via MATLAB® Simulink, which are presented in Section 4.3.2.

After that, a HiL simulation of the stand-alone system and the controller are implemented within the FPGA. In this way, the VHDL implementation of the controller is verified by comparing the FPGA simulations and the MATLAB® simulations. The sampling interval for the HiL simulation is chosen as  $T_{hil} = 1 \mu\text{s}$ , similar to the one used for MATLAB® simulations. For the model predictive controller the sampling interval is  $T_s = 25 \mu\text{s}$ , unless stated otherwise. The clock frequency is set to  $f_{clk} = 20 \text{ MHz}$ .

By extending the prediction horizon, the number of possible solutions increases exponentially. Unavoidably, the number of clock cycles required to compute the optimal solution on the FPGA also increases exponentially. This is illustrated in Figure 5.11. The number of cycles corresponding to a prediction horizon were approximated using a counter during the MATLAB® simulations. For  $N_p = 3$  and  $N_p = 10$ , roughly 14 and 2024 clock cycles, respectively, would be required to complete the online calculations of the non-recursive MPC algorithm within a sampling period.<sup>12</sup>

For the MATLAB® simulations, the longest prediction horizon considered was  $N_p = 10$ . With that being said, the online calculations of the model predictive controller need to be performed on the FPGA within a sampling period in  $T_s f_{clk} = 500$  clock cycles to guarantee optimality. To implement the model predictive controller online for  $N_p = 8$  about 510 clock cycles would be required.<sup>12</sup> Therefore, prediction horizons beyond  $N_p = 7$  are not considered for implementation on the FPGA.<sup>13</sup>

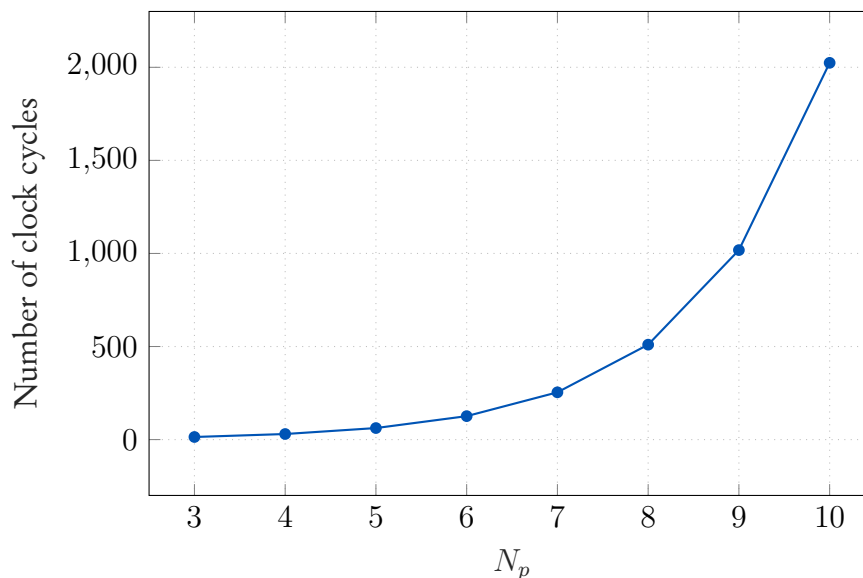


Figure 5.11: Number of clock cycles approximated in MATLAB®, which will be required to obtain the optimal solution from the non-recursive MPC algorithm.

<sup>12</sup>The values are not clearly showing on the figure due the large scale.

<sup>13</sup>Note that longer horizons can be achieved by increasing the sampling interval  $T_s$ , which would require re-computing some matrices offline. Thereafter, the new matrices are loaded onto the FPGA, which is recompiled.

For the implementation of the model predictive controller with  $N_p = 7$ , i.e., the longest horizon considered, only 38 DSP blocks out of the 220 DSP blocks available on the Xilinx FPGA are used. This shows how effectively the resources were managed during the implementation, especially by calculating the matrices offline instead of online. Also, for the online calculation of the initial upper bound cost  $J_{ini}$ , the same multipliers are recycled by calculating only one entry of  $J_{ini}$  at every clock cycle. As a result, the resource usage is reduced significantly, however, at the expense of the time taken to complete the calculations. It can be seen that there is a trade-off between resources usage and the time taken to implement tasks on the FPGA.

### Simulation waveforms

For the prediction horizon of  $N_p = 7$ , the horizon is split in accordance with the move blocking strategy with  $N_1 = 5$ ,  $N_2 = 2$  and  $n_s = 4$ . The penalty matrix and the scalar weighting factor in the cost function are chosen as

$$\mathbf{Q} = \text{diag}(250, 90) \text{ and } \lambda_2 = 10,$$

respectively, with  $\lambda_u$  adjusted to obtain an average switching frequency of  $f_{sw} \approx 16$  kHz. The same parameters were used for the MATLAB<sup>®</sup> simulation, for the same prediction horizon.

Figure 5.12 shows a comparison of the inductor current of the boost converter for the FPGA and MATLAB<sup>®</sup> simulations. It can clearly be seen that the controller algorithm is implemented correctly on the FPGA, as both simulations show similar results. It should also be noted that the simulations are not identical since the controller and the HiL simulation run in parallel with each other on the FPGA. Whereas in MATLAB<sup>®</sup>, they run in a non-parallel manner; the simulation of the stand-alone system is paused when the controller is running, and vice versa. Another difference worth mentioning is that the MATLAB<sup>®</sup> simulation has zero delay compensation, whereas delay compensation is added in the FPGA simulation. Also, in the FPGA implementation, there are rounding errors.

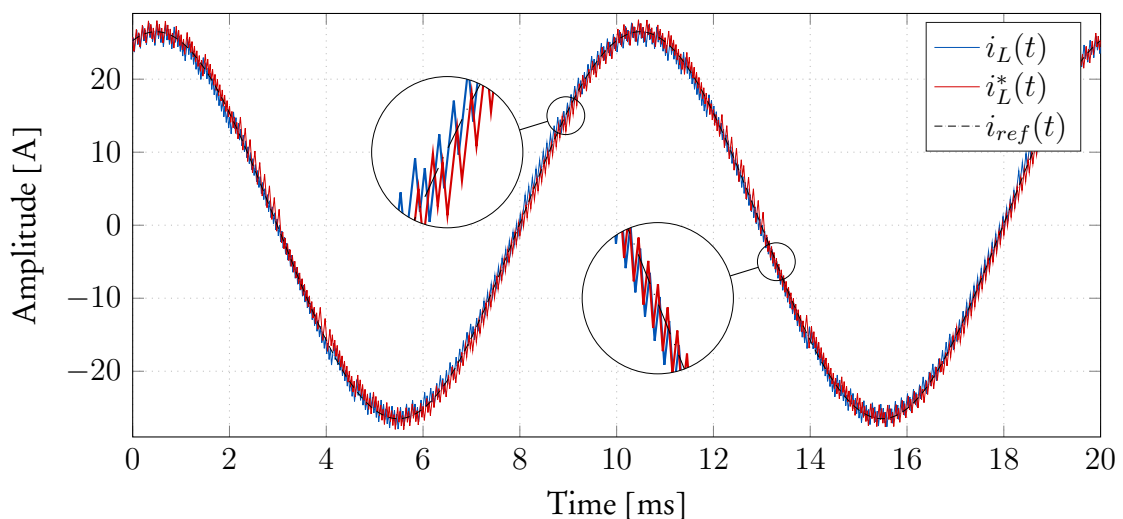


Figure 5.12: FPGA and MATLAB<sup>®</sup> simulations comparison for the inductor current of the boost converter for  $N_p = 7$ . The inductor current for the FPGA and MATLAB<sup>®</sup> simulations are denoted by  $i_L(t)$  and  $i_L^*(t)$ , respectively. The reference ripple current  $i_{ref}(t)$  is shown by the black dash dotted line.

Figure 5.13 shows a comparison of the active capacitor voltage of the boost converter for the FPGA and MATLAB® simulations. It should be apparent that both simulations are similar; however, they are not identical, for the same reasons mentioned before.

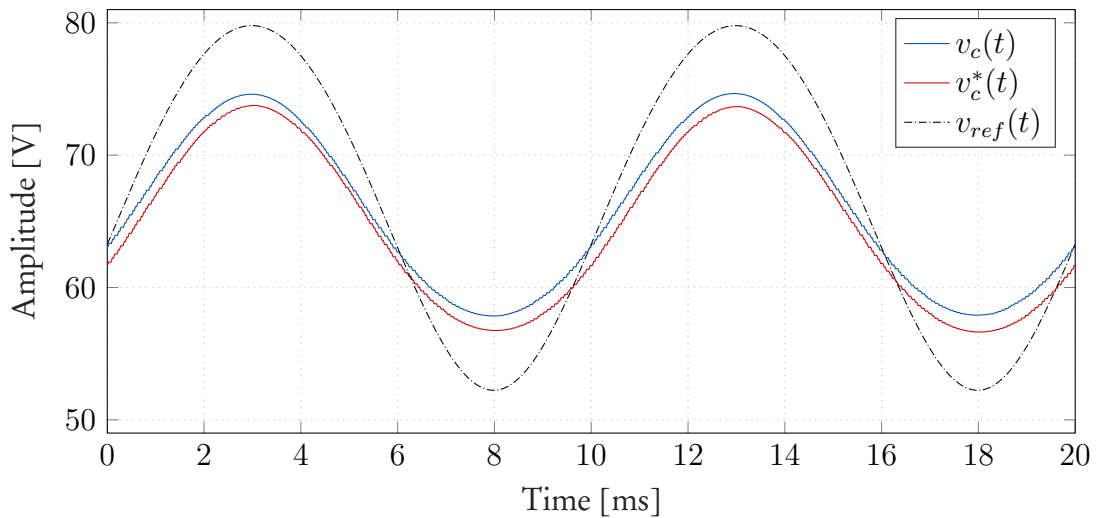


Figure 5.13: FPGA and MATLAB® simulations comparison for the active capacitor voltage of the boost converter for  $N_p = 7$ . The active capacitor voltage for the FPGA and MATLAB® simulations are denoted by  $v_c(t)$  and  $v_c^*(t)$ , respectively. The reference voltage  $v_{ref}(t)$  is shown by the black dash dotted line.

### Harmonic analysis

To verify whether the controller implemented on the FPGA achieved its main objective of reducing the ripple current passing through the battery, an FFT analysis is carried out. The second-order harmonic component  $i_{100\text{Hz}}(t)$  in the battery current  $i_b(t)$  obtained from the HiL simulation is analyzed.

Before the boost converter is turned on the amplitude of  $I_{100\text{Hz}} \approx 26.30$  A, as shown in Figure 5.14. After the boost converter is turned on, the amplitude of  $I_{100\text{Hz}} = 0.85$  A, as shown in Figure 5.15. It should be apparent that the ripple current passing through the battery is significantly reduced to approximately 3.2% of the nominal battery current. Hence, the practical feasibility of the controller design is verified.

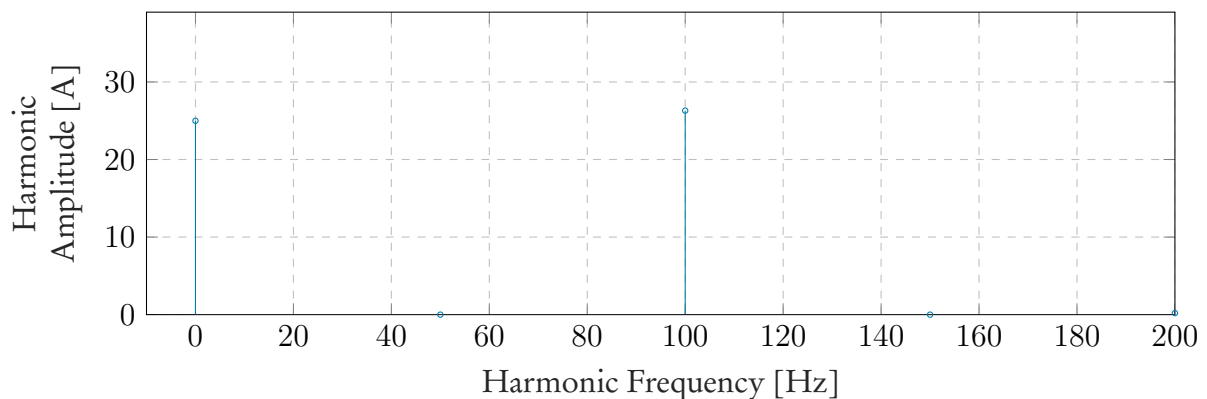


Figure 5.14: Harmonic amplitude spectrum of the current through the battery  $i_b(t)$  for the time before the boost converter is turned on, in the HiL simulation within the FPGA.



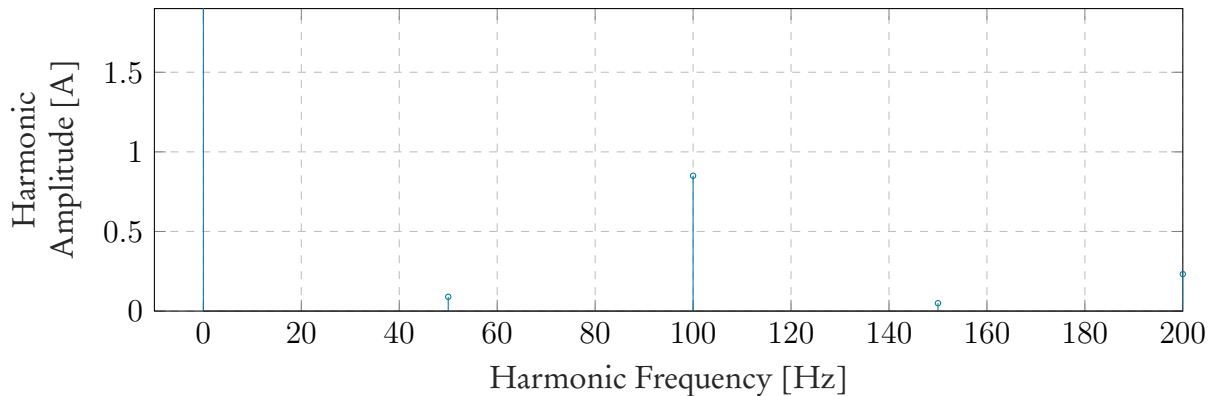


Figure 5.15: Harmonic amplitude spectrum of the current through the battery  $i_b(t)$  for the time after the boost converter is turned on, in the HiL simulation within the FPGA.

### Performance evaluation of long horizons

In order to perform an analysis on the benefits of long prediction horizons while using the practical controller executed on the FPGA, the weighting factor  $\lambda_u$  is adjusted for all the horizons considered. This is done to attain an average switching frequency of 16 kHz. The penalty matrix in the cost function are kept the same as

$$\mathbf{Q} = \text{diag}(250, 90).$$

The simulations performed within the FPGA and in MATLAB<sup>®</sup> for the same horizons are compared. Table 5.1 shows the amplitude of the second-order harmonic component  $I_{100\text{Hz}}$  for all the horizons considered, with  $N_p = 7$ , as the longest horizon considered. Both the FPGA and MATLAB<sup>®</sup> simulations are included in the table.

By analyzing only the results obtained from the FPGA simulations, it can be seen that as the prediction horizon is extended the ripple current passing through the battery is reduced, albeit gradually. For  $N_p = 3$ , the amplitude of the ripple component is slightly higher than that of  $N_p = 7$ , i.e., referring to the one with  $N_1 = 5$  and  $N_2 = 2$  prediction steps.

To analyze the impact of changing the number of prediction steps of the split-horizon in accordance with the move blocking strategy,  $N_p = 7$  is used as an example. It is apparent that increasing the number of steps of the second part of the horizon, which are sampled more coarsely, reduced the performance of the controller. Hence, it is recommended to have more prediction steps for the first part of the horizon, with  $N_1$  steps, to ensure high-resolution sampling of the system.

By comparing both the FPGA and MATLAB<sup>®</sup> simulation results, it can be seen that the controller performed better for the MATLAB<sup>®</sup> simulations, as the amplitude of the second-order harmonic component is much smaller than that of the FPGA simulations for the various horizons considered. Nonetheless, the MATLAB<sup>®</sup> simulations represent the controller based on a theoretical model. Whereas, the FPGA simulations best describe the real behaviour of the controller in a practical system. However, in both simulations, the control objective is met since the ripple current is reduced to less than 10 % of the nominal battery current for all the horizons considered.

Table 5.1: FPGA and MATLAB® simulations comparison of the second-order harmonic component  $I_{100\text{Hz}}$  at  $f_{sw} \approx 16\text{ kHz}$  for various prediction horizons.

Horizon	$N_1 + N_2$	MATLAB®			FPGA		
		$\lambda_u$	$f_{sw}$ [kHz]	$I_{100\text{Hz}}$ [A]	$\lambda_u$	$f_{sw}$ [kHz]	$I_{100\text{Hz}}$ [A]
$N_p = 3$	2 + 1	10	17.385	1.21	5	17.601	1.65
$N_p = 4$	3 + 1	20	16.005	0.67	25	16.101	1.00
$N_p = 5$	3 + 2	10	15.995	0.69	45	16.101	1.04
$N_p = 6$	4 + 2	30	16.050	0.67	150	16.101	0.98
$N_p = 7$	5 + 2	27	16.100	0.70	250	16.301	0.85
$N_p = 7$	4 + 3	25	16.001	0.73	150	16.051	1.16

## 5.4 Summary

Due to the computational complexity of the proposed non-recursive MPC algorithm, an FPGA was chosen based on its increased processing power. In this chapter, a detailed description of the architecture and physical capabilities of the FPGA board used was given. A few arithmetic concepts concerning VHDL were discussed. Also, an insight into the notion of computational delay compensation was discussed since, in the previous chapters, the idealized case of predictive control was assumed.

The VHDL implementation of the proposed model predictive controller, which comprises the initial upper bound cost calculation algorithm and the non-recursive MPC algorithm, was discussed in great detail. The usage resources on the FPGA was significantly reduced by structuring the code efficiently to ensure the possibility of implementing longer prediction horizons. To verify the practical feasibility of the controller design, a HiL simulation of the stand-alone system was conducted within the FPGA. Furthermore, the implementation of PWM with unipolar switching was explained for controlling the single-phase dc-to-ac converter in the HiL simulation.

The simulations were done for different prediction horizons that can achieve steady-state operation, i.e., prediction horizons  $N_p > 2$ . The longest horizon considered was  $N_p = 7$ . The control objective was met since the ripple current passing through the battery was reduced significantly to roughly 3.2% of the nominal battery current with the practical controller on the FPGA. It can be concluded that ripple energy compensation in single-phase dc-to-ac converters using a boost converter with long-horizon direct MPC is possible in a practical system.

# Chapter 6

## Conclusions

This chapter concludes the thesis. The main results obtained from the relevant chapters are summarized. Recommendations for possible future work concerning the research conducted are discussed.

### 6.1 Overview of results

To recap the primary objective of this thesis: a dc-to-dc boost converter was connected on the dc side of a single-phase dc-to-ac converter, in parallel with a battery energy storage system, for ripple energy compensation. The boost converter semiconductor switches needed to be manipulated so that the ripple current is directed away from the battery to the active capacitor of the boost converter, which introduced a control problem. Thus, a control strategy based on long-horizon direct MPC had to be formulated and verify its feasibility through simulations in MATLAB<sup>®</sup> and practically on an FPGA.

#### 6.1.1 Performance evaluation of long prediction horizons

The formulated non-recursive MPC algorithm that incorporates both the branch-and-bound technique and move blocking strategy, as optimization methods, was successfully implemented in MATLAB<sup>®</sup>. The optimal solution of the control algorithm was used to manipulate the switch positions of a boost converter to compensate for ripple energy in a battery-powered single-phase dc-to-ac converter that is either connected to the grid or a load (stand-alone). The pulsating power transferred from the ac side onto the dc side of the single-phase dc-to-ac converter generates second-order harmonic currents through the dc bus. Therefore, the boost converter was controlled reduce the generated ripple current passing through the battery so that it does not exceed 10 % of the nominal battery current. Note that the single-phase dc-to-ac converter was controlled using PWM with unipolar switching, an open-loop controller.

Due to the non-minimum phase behaviour that boost converters exhibit, controlling the active capacitor voltage required long prediction horizons so that the controller could predict beyond the initial voltage drop when increasing the voltage across the active capacitor. For prediction horizons  $N_p < 3$ , the system is unstable, which validated the claim that longer horizons provide performance benefits.

A diverse set of scalar weights  $\lambda_u$  were considered for different prediction horizons, from  $N_p = 3$  to  $N_p = 10$ , to attain an average switching frequency of 16 kHz. For both the grid-connected system and the stand-alone system, the inductor current of the boost converter closely tracked the given reference ripple current. Thus, one of the primary control objectives

was met. Ideally, the active capacitor voltage was supposed to track its given time-varying voltage reference closely. However, it was too sluggish to follow its reference. One reason for this behaviour was attributed to the voltage reference not being entirely accurate due to the assumptions made during its derivation. Also, since the control of the inductor current was prioritized more than that of the active capacitor voltage, the voltage tracking error was penalized less heavily than the current tracking error in the formulated control law. As a result, this might have affected the control of the active capacitor voltage.

Nonetheless, a gradual decrease in the ripple current passing through the battery was observed as the prediction horizon was extended. By analyzing the horizons,  $N_p = 3$  and  $N_p = 10$ , the shortest and longest horizons considered; for the grid-connected system, the ripple current was reduced to approximately 4.0 % and 2.4 % of the nominal battery current, respectively; for the stand-alone system, the ripple current was reduced to approximately 4.6 % and 2.8 % of the nominal battery current, respectively. In both systems, the ripple current was reduced to less than the maximum allowable ripple current to flow through the battery. Furthermore, it can be seen that longer prediction horizons did offer significant performance benefits.

The response time of the controller during transients was examined for the prediction horizon of  $N_p = 10$  using the stand-alone system. It was shown that the controller reacted quickly to a step-change in the load current with a settling time in the order of 10 ms.

It should be noted that a significant reduction in the capacitance requirements is noticed for both systems. In the grid-connected system, the capacitance required was reduced by 95.5 %, and in the stand-alone system, it was reduced by 97.5 %.

### 6.1.2 FPGA implementation

To ensure the practical feasibility of the non-recursive MPC algorithm, a HiL simulation that assumes the role of a practical system had to be executed on an FPGA. The controller and the HiL simulation were successfully implemented on the Xilinx XC7Z020-1CLG484C Zynq-7020 FPGA. The controller comprises the pulse-width modulator and the model predictive controller, which constitutes an algorithm that computes the initial upper bound cost and the non-recursive MPC algorithm. Out of the 220 DSP blocks on the host FPGA, only 38 DSP blocks were used to implement the model predictive controller by carefully structuring the code.

A verification HiL simulation of the stand-alone system was implemented within the FPGA with  $N_p = 7$  as the longest prediction horizon considered. Using  $N_p = 7$ , it was demonstrated that the FPGA simulation waveforms of the inductor current and active capacitor voltage of the boost converter are very similar to the MATLAB® simulation waveforms, with regards to following their given references.

However, it was shown that the FPGA and MATLAB® simulation waveforms were not exactly identical due to implementation differences. On the FPGA, the HiL simulation and the controller run in parallel with each other. Conversely, in MATLAB®, when the controller is running, the simulation of the stand-alone system is paused, and vice versa. Moreover, for the FPGA implementation, rounding errors exist.

To evaluate the performance benefits of long horizons on the FPGA, simulations were conducted for different prediction horizons. It was observed that as the prediction horizon was extended, the ripple current passing through the battery decreased, although gradually. For  $N_p = 3$  and  $N_p = 7$ , the shortest and longest horizons considered, the ripple current was reduced to roughly 6.3 % and 3.2 % of the nominal battery current, respectively.

The FPGA and MATLAB® simulations were compared for different prediction horizons. The comparison showed that, generally, the MATLAB® simulations performed slightly better than the FPGA simulations in terms of ripple energy compensation. This results from MATLAB® simulations being performed using an idealized setup without any second-order effects, such as computational delays in the controller implementation.

## 6.2 Recommendations for future research

### 6.2.1 Tuning of parameters

A discussion that gives recommendations on the tuning of parameters that affect the system performance, such as the weighting factors and sampling interval, is presented.

#### Tracking error

The ability of MPC with reference tracking to incorporate several system variables into a single cost function and control them concurrently is one of its main advantages. A scalar weight that is specific to each tracking error term sets the relative importance between the system variables. To be more specific, in this thesis, the boost converter control prioritized the inductor current by penalizing its tracking error more heavily, as it is considered a primary control objective. It was penalized using the scalar weight  $\lambda_1$ . Conversely, less weight was imposed on the scalar weight  $\lambda_2$ , which penalizes the active capacitor voltage tracking error. To achieve the desired system performance,  $\lambda_1$  and  $\lambda_2$  were determined empirically on a trial-and-error basis.

The empirical method of obtaining the scalar weights used in this thesis was questionable, and in general, the method is ambiguous and has known limitations [36]. It does not provide a systematic approach to settling the trade-off present in designing the weighting factors. As a result, it poses challenges in designing the control system. For example, during simulations, it seemed as if the inductor current and active capacitor voltage reference signals contradicted each other. The reason for that is, whenever the current tracking error was penalized more heavily, the voltage error did not entirely minimize. Conversely, when the voltage error was prioritized by penalizing it more heavily than the current error, the inductor current struggled to follow its reference signal.

The contradiction could be attributed to the overestimation of the importance of the inductor current control compared to that of the active capacitor voltage, and as a result, the active capacitor voltage control became challenging. This implies that the two controlled variables could be of equal importance. Also, the difference in the physical nature of the inductor current and active capacitor voltage could have led to coupling effects between the two system variables [30].

It is recommended that the weighting factors be obtained using analytical or numerical methods since they profoundly influence system performance. In [30], a set of guidelines are proposed that employ a branch-and-bound method to reduce the uncertainty of choosing the weighting factors. In [37], algebraic design guidelines of computing the optimal weighting factor corresponding to each system variable in a cost function are proposed.

### Switching effort

The weighting factor  $\lambda_u$  that adjusts the switching frequency is also chosen empirically. For a given horizon, the operating point changes; hence,  $\lambda_u$  had to be adjusted to attain the desired switching frequency. Due to this, the process became monotonous. Thus, it is recommended to use an analytical method to appropriately select the weight on the switching effort, similar to the analytical expression derived in [37]. At the time of writing this thesis, a suggestion given in [36] is to avoid adjusting the controlling effort altogether and using a fixed switching frequency predefined by the sampling interval.

Throughout the simulations carried out in this thesis, only  $\lambda_u$  was used as the tuning parameter to adjust the switching frequency. However, the sampling interval also influences the closed-loop performance of the system. It is recommended that a combination of random sampling intervals and weighting factors with a wide range be considered to investigate the performance of the system. A similar approach, referred to as the Monte Carlo simulations, are used in [31].

By using different sampling intervals, prediction horizons longer than  $N_p = 7$  could be implemented within the FPGA. Hence, further analysis of the performance of long prediction horizons on a practical system could be evaluated.

### 6.2.2 Active capacitor voltage reference

As mentioned in the previous section, the active capacitor voltage was too sluggish to follow the given time-varying voltage reference. It is believed that another reason for this is associated with the initial assumption made when deriving the time-varying voltage reference.

It is assumed that all of the positive ripple current flowing into the boost converter charges the active capacitor for the derivation. This assumption does not always hold due to the switch  $S_1$  between the inductor and the active capacitor. When the control input  $u = 0$ , the switch  $S_1$  is off, and the positive ripple current flows through switch  $S_2$ . The same is true when the ripple current is negative, and the active capacitor is discharging through the dc bus.

Thus, it is recommended to derive a reference for the active capacitor voltage that considers the switching transitions of the boost converter semiconductor switches.

### 6.2.3 Experimental tests

Since the practical feasibility of long-horizon direct MPC for ripple energy compensation in a single-phase dc-to-ac converter was only verified through a HiL simulation on an FPGA, experimental tests on a physical hardware setup should be performed.

It would be interesting also to implement the proposed control strategy on more complex systems. For example, a cascaded h-bridge multilevel converter, with each converter module connected to a boost converter in parallel with the dc supply voltage. This could be implemented for ripple energy compensation in the multilevel converter.

# Appendices

# Appendix A

## Mathematical preliminaries

Some of the mathematical terms and definitions used in this thesis are briefly explained.

### The norm relative to a vector

The magnitude of a vector is referred to as the *vector norm*. Given a vector

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \end{bmatrix},$$

the vector norm  $\|\mathbf{x}\|$  is non-negative real value. The 2-norm (or Euclidean norm) of the vector is defined as

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^n x_i^2}.$$

A weighted Euclidean norm of a vector

$$\|\mathbf{x}\|_Q^2 = \mathbf{x}^T \mathbf{Q} \mathbf{x},$$

where  $\mathbf{Q}$  denotes a positive definite matrix.

### Positive definite matrix

In linear algebra, a  $n_r \times n_r$  matrix  $\mathbf{Q}$  (i.e., a symmetric matrix) is considered *positive definite* if

$$\mathbf{x}^T \mathbf{Q} \mathbf{x} > 0,$$

for every non-zero column vector  $\mathbf{x} \in \mathbb{R}^{n_r}$ . Note that  $\mathbf{x}^T$  denotes the transpose of the vector  $\mathbf{x}$ . Also, the positive definite matrix  $\mathbf{Q}$  must have all positive eigenvalues.

A quadratic form can be used to check positive definiteness of a matrix, for example: Let  $f(\mathbf{x}) = \mathbf{x}^T \mathbf{Q} \mathbf{x}$ , where

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \text{and} \quad \mathbf{Q} = \begin{bmatrix} a & b \\ b & c \end{bmatrix}. \quad (\text{A.1})$$

The quadratic form is  $f(\mathbf{x}) = ax_1^2 + 2bx_1x_2 + cx_2^2$ . If the quadratic form is positive for every (real)  $x_1$  and  $x_2$ , then the matrix  $\mathbf{Q}$  is positive definite.



## Appendix B

### Ripple power analysis of a single-phase dc-to-ac converter with an $RL$ load

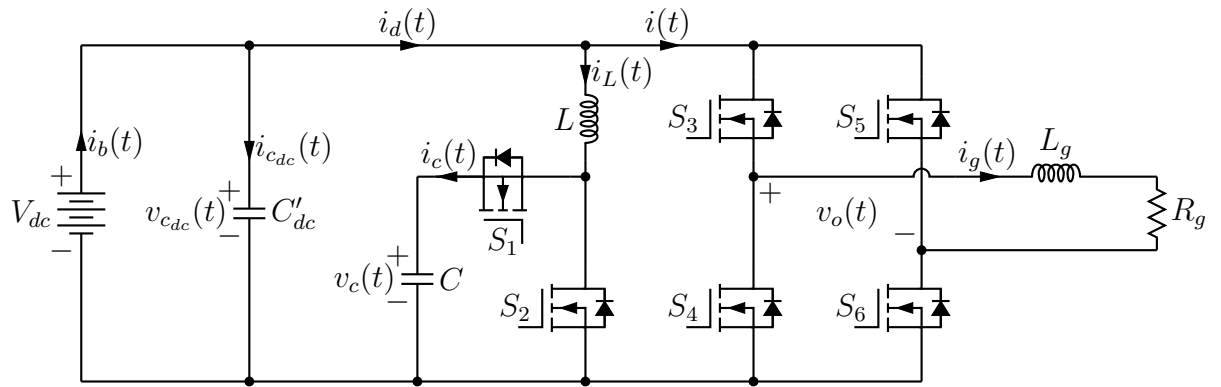


Figure B.1: Single-phase dc-to-ac converter with an  $RL$  load and a dc-to-dc boost converter connected on the dc side, in parallel with the battery energy storage system.

Consider the single-phase dc-to-ac converter with an  $RL$  load, denoted by  $R_g$  and  $L_g$ , as shown in Figure B.1. The instantaneous power injected into the load side

$$p_g(t) = i_g^2(t)R_g, \quad (\text{B.1})$$

with the load current

$$i_g(t) = I_g \sin(\omega_1 t - \phi), \quad (\text{B.2})$$

where  $I_g$  is the amplitude of the load current,  $\omega_1 = 2\pi f_1$ , with  $f_1$  as the fundamental frequency and  $\phi$  is the phase angle.

The instantaneous power on the load side is given by

$$p_g(t) = \frac{I_g^2 R_g}{2} \cos(\phi) - \frac{I_g^2 R_g}{2} \cos(2\omega_1 t - \phi). \quad (\text{B.3})$$

It can be noted that in (B.3), the instantaneous power consists of two terms: the *dc component* and the undesirable *second-order harmonic component*.

The dc component is denoted by

$$P_o = \frac{I_g^2 R_g}{2} \cos(\phi), \quad (\text{B.4})$$

and the second-order harmonic component, which is the ripple power is denoted by

$$p_r(t) = -\frac{I_g^2 R_g}{2} \cos(2\omega_1 t - \phi). \quad (\text{B.5})$$

The ripple power pulsates at twice the fundamental frequency  $f_1$ . Inevitably, the ripple power is transferred from the load side to the dc side and generates second-order harmonic currents, referred to as the ripple current, through the dc bus. The ripple current flows through the battery, and as a result causes reliability issues, as explained in Section 2.1.4.

# Bibliography

- [1] S. Vazquez, S. M. Lukic, E. Galvan, L. G. Franquelo, and J. M. Carrasco, "Energy Storage Systems for Transport and Grid Applications," *IEEE Transactions on Industrial Electronics*, vol. 57, pp. 3881–3895, Dec. 2010.
- [2] J.-M. Tarascon and M. Armand, "Issues and challenges facing rechargeable lithium batteries," *Nature*, vol. 414, pp. 359–367, Nov. 2001.
- [3] J. M. Carrasco, L. G. Franquelo, J. T. Bialasiewicz, E. Galvan, R. C. PortilloGuisado, M. A. M. Prats, J. I. Leon, and N. Moreno-Alfonso, "Power-Electronic Systems for the Grid Integration of Renewable Energy Sources: A Survey," *IEEE Transactions on Industrial Electronics*, vol. 53, pp. 1002–1016, June 2006.
- [4] R. Wang, F. Wang, D. Boroyevich, R. Burgos, R. Lai, P. Ning, and K. Rajashekara, "A High Power Density Single-Phase PWM Rectifier With Active Ripple Energy Storage," *IEEE Transactions on Power Electronics*, vol. 26, pp. 1430–1443, May 2011.
- [5] H. Wen, W. Xiao, X. Wen, and P. Armstrong, "Analysis and Evaluation of DC-Link Capacitors for High-Power-Density Electric Vehicle Drive Systems," *IEEE Transactions on Vehicular Technology*, vol. 61, pp. 2950–2964, Sept. 2012.
- [6] M. Su, P. Pan, X. Long, Y. Sun, and J. Yang, "An Active Power-Decoupling Method for Single-Phase DC-AC Converters," *IEEE Transactions on Industrial Informatics*, vol. 10, pp. 461–468, Feb. 2014.
- [7] X. Cao, Q. Zhong, and W. Ming, "Ripple Eliminator to Smooth DC-Bus Voltage and Reduce the Total Capacitance Required," *IEEE Transactions on Industrial Electronics*, vol. 62, pp. 2224–2235, Apr. 2015.
- [8] Q. Zhong, W. Ming, X. Cao, and M. Krstic, "Control of Ripple Eliminators to Improve the Power Quality of DC Systems and Reduce the Usage of Electrolytic Capacitors," *IEEE Access*, vol. 4, pp. 2177–2187, 2016.
- [9] Y. Sun, Y. Liu, M. Su, W. Xiong, and J. Yang, "Review of Active Power Decoupling Topologies in Single-Phase Systems," *IEEE Transactions on Power Electronics*, vol. 31, pp. 4778–4794, July 2016.
- [10] H. Wang and F. Blaabjerg, "Reliability of Capacitors for DC-Link Applications in Power Electronic Converters-An Overview," *IEEE Transactions on Industry Applications*, vol. 50, pp. 3569–3578, Sept. 2014.
- [11] T. Geyer, *Model Predictive Control of High Power Converters and Industrial Drives*. Chichester, UK: John Wiley & Sons, Ltd.

- [12] M. D. Dorfling, *Practical implementation of long-horizon direct model predictive control*. Thesis, Stellenbosch : Stellenbosch University, Mar. 2018.
- [13] P. Karamanakos, T. Geyer, and S. Manias, "Direct Voltage Control of DC-DC Boost Converters Using Enumeration-Based Model Predictive Control," *IEEE Transactions on Power Electronics*, vol. 29, pp. 968–978, Feb. 2014.
- [14] P. Karamanakos, T. Geyer, N. Oikonomou, F. D. Kieferndorf, and S. Manias, "Direct Model Predictive Control: A Review of Strategies That Achieve Long Prediction Intervals for Power Electronics," *IEEE Industrial Electronics Magazine*, vol. 8, pp. 32–43, Mar. 2014.
- [15] 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.
- [16] M. Chingwena, T. Mouton, and M. Dorfling, "Model Predictive Control of an Active Capacitor for Ripple Energy Compensation in Single-Phase DC-to-AC Converters," in *2019 21st European Conference on Power Electronics and Applications (EPE '19 ECCE Europe)*, pp. P.1–P.10, Sept. 2019. ISSN: null.
- [17] Nihal Kularatna author, *Power electronics design handbook: low-power components and applications*. EDN series for design engineers, Boston: Newnes, 1998.
- [18] D. G. Holmes and T. A. Lipo, *Pulse Width Modulation for Power Converters: Principles and Practice*. John Wiley & Sons, Oct. 2003.
- [19] N. Mohan, T. M. Undeland, and W. P. Robbins, *Power electronics: converters, applications, and design*. Wiley, Jan. 1995.
- [20] J. Soomro, T. D. Memon, and M. A. Shah, "Design and analysis of single phase voltage source inverter using Unipolar and Bipolar pulse width modulation techniques," in *2016 International Conference on Advances in Electrical, Electronic and Systems Engineering (ICAEEES)*, pp. 277–282, Nov. 2016. ISSN: null.
- [21] D. Barater, G. Buticchi, A. S. Crinto, G. Franceschini, and E. Lorenzani, "Unipolar PWM Strategy for Transformerless PV Grid-Connected Converters," *IEEE Transactions on Energy Conversion*, vol. 27, pp. 835–843, Dec. 2012.
- [22] M. Hilmy, M. E. Ahmed, M. Orabi, M. A. Sayed, and M. El-Nemr, "Optimum design of high efficiency power conditioning wind energy system," in *2010 IEEE International Conference on Power and Energy*, pp. 611–616, Nov. 2010.
- [23] L. Gu, X. Ruan, M. Xu, and K. Yao, "Means of Eliminating Electrolytic Capacitor in AC/DC Power Supplies for LED Lightings," *IEEE Transactions on Power Electronics*, vol. 24, pp. 1399–1408, May 2009.
- [24] M. T. Chingwena and H. d. T. Mouton, "A Multilevel Cascaded Converter for a Battery Energy Storage System," in *2019 Southern African Universities Power Engineering Conference/Robotics and Mechatronics/Pattern Recognition Association of South Africa (SAUPEC/RobMech/PRASA)*, pp. 553–558, Jan. 2019.

- [25] Q.-C. Zhong, T. Hornik, “Control of Power Inverters in Renewable Energy and Smart Grid Integration ebook by Qing-Chang Zhong.”
- [26] P. Karamanakos, *Model predictive control strategies for power electronics converters and AC drives*. Phd thesis, 2013.
- [27] E. F. Camacho and C. B. Alba, *Model Predictive Control*. Advanced Textbooks in Control and Signal Processing, London: Springer-Verlag, 2 ed., 2007.
- [28] L. Grüne and J. Pannek, *Nonlinear Model Predictive Control: Theory and Algorithms*. Communications and Control Engineering, London: Springer-Verlag, 2011.
- [29] T. Dorfling, H. d. T. Mouton, T. Geyer, and P. Karamanakos, “Long-Horizon Finite-Control-Set Model Predictive Control With Nonrecursive Sphere Decoding on an FPGA,” *IEEE Transactions on Power Electronics*, vol. 35, pp. 7520–7531, July 2020. Conference Name: IEEE Transactions on Power Electronics.
- [30] P. Cortes, S. Kouro, B. La Rocca, R. Vargas, J. Rodriguez, J. I. Leon, S. Vazquez, and L. G. Franquelo, “Guidelines for weighting factors design in Model Predictive Control of power converters and drives,” in *2009 IEEE International Conference on Industrial Technology*, pp. 1–7, Feb. 2009.
- [31] 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, Mar. 2015. Conference Name: IEEE Transactions on Power Electronics.
- [32] Xilinx, “Zynq-7000 SoC Data Sheet: Overview (DS190),” p. 25, 2018.
- [33] P. D. Cem Ünsalan and P. D. Bora Tar, *Programming the FPGA*. McGraw-Hill Education: New York, Chicago, San Francisco, Athens, London, Madrid, Mexico City, Milan, New Delhi, Singapore, Sydney, Toronto, 2017. Book Title: Digital System Design with FPGA: Implementation Using Verilog and VHDL.
- [34] Xilinx, “Logicore ip integrated logic analyzer (ila) v2.1,” 2013.
- [35] K. Ogata, *Modern Control Engineering*. Boston: Pearson, 5 edition ed., Sept. 2009.
- [36] P. Karamanakos and T. Geyer, “Guidelines for the Design of Finite Control Set Model Predictive Controllers,” *IEEE Transactions on Power Electronics*, vol. 35, pp. 7434–7450, July 2020. Conference Name: IEEE Transactions on Power Electronics.
- [37] T. Geyer, “Algebraic Tuning Guidelines for Model Predictive Torque and Flux Control,” *IEEE Transactions on Industry Applications*, vol. 54, pp. 4464–4475, Sept. 2018. Conference Name: IEEE Transactions on Industry Applications.