Model Predictive Control with Space-Vector Modulation for a Grid-Connected Converter with an *LCL*-Filter

by

Adriaan Sadie

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 2020

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.

March 2020

Abstract

This thesis presents an indirect model predictive control (MPC) strategy with space-vector modulation (SVM). The controller is used to control a high-power voltage-source inverter (VSI), connected to the grid via an *LCL*-filter. The controller evaluates a multi-variable and convex cost function over a long prediction horizon in order to determine an optimal sequence of references for a space-vector modulator. The MPC strategy is tested in a MATLAB simulation. The results of the simulation show excellent steady-state behaviour as well as a fast response during transients. The controller successfully dampens the resonant frequency of the filter and exhibits low grid-current harmonic distortion.

The controller is then tested by means of a hardware-in-the-loop (HiL) simulation to verify the functionality of the controller on a field-programmable gate array (FPGA) device. The VHDL (Very high-speed integrated circuit Hardware Description Language) design of the controller is discussed in detail, where emphasis is put on the FPGA resource and time usage. To confirm the efficacy of the design of the controller, the HiL results are compared to MATLAB equivalent simulation results. The signals from the HiL compare well to the MATLAB results, with only a slight offset, due to differences in the PWM pulses produced by the FPGA-based controller. The controller is able to function in real-time on the FPGA.

An alternative small-signal linear quadratic regulator (LQR) controller is also investigated in this thesis. The controller is first tested by means of a MATLAB based simulation, which shows excellent steady-state behaviour with low Total Harmonic Distortion (THD) and fast settling time during transients. The controller is also implemented on an FPGA and tested by means of a HiL simulation. The results of the HiL and the MATLAB signals are compared, and the LQR controller is able to function in real-time on the FPGA.

Opsomming

Hierdie tesis bied 'n indirekte modelvoorspelling-beheerstrategie met ruimtevektor-modulasie. Die beheerder word gebruik om 'n hoë-drywing spanningsbron-omsetter aan te dryf, wat gekoppel is aan die kragnetwerk. Die beheerder evalueer 'n multiveranderlike en konvekse kostefunksie oor 'n lang voorspellingshorison, om sodoende 'n optimale volgorde van verwysingseine vir 'n ruimtevektormodulator te bepaal. Die modelvoorspelling-beheerstrategie word getoets deur middel van 'n MATLAB simulasie. Die resultate van die simulasie wys uitstekende bestendigetoestand-optrede asook 'n vinnige reaksie tydens oorgangsverskynsels. Die beheerder is in staat om die resonante frekwensie van die filter te demp en toon lae kragnetwerk stroomharmoniese distorsie.

Die beheerder word dan getoets deur middel van 'n hardeware-in-die-lus (HiL) simulasie om die funksionaliteit van die beheerder op die (veldprogrammeerbare hekskikking) FPGA te bevestig. Die (hoëspoed geïntegreerde stroombaan hardeware beskrywingstaal) VHDL-ontwerp van die beheerder word volledig bespreek waar klem geplaas word op die FPGA se hulpbron- en tydsverbruik. Om die effektiwiteit van die beheerderontwerp te bevestig, word die HiL resultate vergelyk met die MATLAB ekwivalente uitslae. Die seine van die HiL vergelyk goed met die MATLAB-resultate, waar slegs 'n klein afwyking waargeneem word. Die afwyking word toegeskryf aan 'n verskil in die pulswydte modulasie-pulse (PWM) van die FPGA beheerder teenoor díe van die MATLAB seine. Die beheerder is in staat om in-tyd te funksioneer op die FPGA.

'n Alternatiewe kleinsein lineêre kwadratiese reguleerder (LKR) beheerder word ook ondersoek in hierdie tesis. Die beheerder word eers getoets deur middel van MATLAB gebaseerde simulasies. Die resultate van die simulasie wys uitstekende bestendigetoestand-optrede, 'n lae totale harmoniese versteuring asook 'n vinnige reaksie tydens oorgangsverskynsels. Die LKR-beheerder word ook getoets op 'n FPGA deur middel van 'n HiL-simulasie. Die uitslae van die HiL en die MATLAB-simulasie word vergelyk, en daar word bevind dat die LKR-beheerder in-tyd kan funksioneer op die FPGA.

Acknowledgements

I would firstly like to thank my God for gifting me with the opportunity to pursue my master's degree. The road was long, but by His grace I managed to survive this daunting task.
For the last three years, I had the opportunity to have Prof Toit Mouton as my study leader, first for skripsie and now for my master's degree. I would like to thank Prof Mouton for guiding me throughout this journey, always having advice, knowledge, and humour at the ready.
I had the privilege of befriending Tinus Dorfling, who also became a mentor to me. Thank you for all the laughs, the walks to De Vrije Burger and the Neelsie, and always being willing to help when the VHDL got the better of me.
I had the opportunity of working with Tobias Geyer the past year. Thank you for always responding quick to queries, giving me extensive feedback on the conference paper, and for your amazing textbook.
I would like to thank my wife and best friend, Martie, for carrying me through the hard times the past two years. Your constant support and love means the world to me, and I look forward to our journey together.
Lastly I would like to thank my parents, Johan and Karen, for believing in me and making it possible for me to finish my studies without having to worry about finances. I appreciate the sacrifices you made for me.

Publication

A. Sadie, H. du Toit Mouton, M. Dorfling and T. Geyer: "Model-predictive control with space-vector modulation for a grid-connected converter with an LCL-filter", Proc. of the IEEE European Conference on Power Electronics and Applications, Genova, Italy, Sept. 2019.

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

AC	alternating current
ADC	analogue-to-digital converter
ARM	advanced RISC machine
CB-PWM	carrier-based pulse-width-modulation
DC	direct current
DSP	digital signal processor
ESR	equivalent series resistance
FPGA	field-programmable gate array
FPU	floating point unit
HiL	hardware-in-the-loop
IDE	integrated development environment
IEC	International Electrotechnical Commission
IGBT	insulated-gate bipolar transistor
LCL	inductor-capacitor-inductor
LQR	linear quadratic regulator
LUT	lookup table
MIMO	multi-input multi-output
MLC	multi-loop control
MPC	model-predictive control
MB	megabyte
NERSA	National Energy Regulator of South Africa
PCC	point of common coupling
PI	proportional-integral
PID	proportional-integral-derivative

PLL	phase-locked-loop
pu	per unit
PWA	piecewise affine
PWM	pulse-width modulation
QP	quadratic program
RAM	random access memory
rms	root-mean-squared
SISO	single-input single-output
SVM	space-vector modulation
THD	total harmonic distortion
VHDL	VHSIC hardware description language
VHSIC	very high-speed integrated circuit
VR	virtual resistance
VSI	voltage-source inverter
ZoH	zero-order-hold

Chapter 1

Introduction

1.1 Background

The field of power electronics has seen an increase in the popularity of model predictive control (MPC) during the last couple of years. Traditionally, MPC has been limited to chemical plants because of the high computational burden when solving the underlying optimisation problem. With the advancement of high-speed computational devices over the last few years such as field-programmable gate arrays (FPGA), MPC could be applied to numerous other applications.

This thesis will focus on indirect MPC, where there is a modulator between the controller and the converter. The advantage of this approach is that the optimisation problem is convex, since the cost function and the constraints are convex. This thesis will also investigate a trajectory-based linear quadratic regulator (LQR) method.

Grid-connected converter applications must meet the relevant grid standards. In this regard, *LCL*-filters are very popular, because of their high attenuation properties compared to the series inductor filters. One aspect that has to be kept in mind when designing an *LCL*-filter is the damping of the resonant frequency of the filter. The two main strategies used to address the resonant frequencies are known as passive and active damping. The former is quite easy to implement but has the drawback of adding damping resistors to the circuit which increase the ohmic losses greatly. There are a few papers on MPC methods that consider active damping. In [2], a term is included in the cost function that results in a high gain at the resonant frequency. This causes a large weight on the harmonic component when close to that frequency. Another method, examined in [3] and [4], implements a virtual damping resistance by injecting an additional current component on the converter-side. In [5], a reduced model of the *LCL*-filter is used to limit the number of sensors required for the controller, and an active damping strategy is also applied. The main advantage of the proposed controller in this thesis, when compared to the previously mentioned strategies, is that it automatically avoids the resonance frequency of the filter without any active of passive damping techniques.

Similar applications of MPC for two-level voltage-source inverters (VSI) with *LCL*-filters without active damping are also considered, such as [6], [7]. In [6], the pulse-width modulator is approximated as a piecewise-affine (PWA) function, which results in a PWA model of the overall system. The optimal control problem can be solved off-line using

explicit MPC. As a result, the state-space is partitioned into a set of polyhedra, where each one is associated with a PWA control law. This approach is limited to short horizons and uses a high switching frequency in relation to the resonant frequency of the filter. In contrast, the prediction horizon of the proposed controller in this thesis can be increased without substantially increasing the problem size and consequently, the computational complexity. In [7], a long prediction horizon is incorporated for an application where the converter is connected to a load via an LC-filter. This approach also uses a continuous MPC strategy in conjunction with a pulse-width modulator. One drawback in [7] is the high switching frequency of 10 kHz. The proposed controller in this thesis is able to run at a relatively low switching frequency of 1.65 kHz.

This thesis proposes a multi-variable model predictive controller with a relatively low switching frequency and a long prediction horizon. These traits help the controller to damp the resonant frequency and ensures stability during transients [4]. The proposed controller is used to control a VSI connected to the grid via an LCL-filter. The main advantage of the control strategy is that it allows the converter to operate at a switching frequency close to the resonant frequency of the LCL-filter. When comparing the proposed strategy to a typical PI based controller, which requires a switching frequency of at least twice the resonant frequency of the filter, the proposed controller seems to be an attractive alternative.

1.2 Thesis Objectives

The three main objectives for this thesis are:

- Development of an indirect MPC strategy that can operate close or below the resonant frequency of the LCL-filter.
- Implementation and verification of such a controller on a low-cost FPGA device, by means of a hardware-in-the-loop (HiL) simulation.
- Investigation and implementation of a small-signal LQR controller, as developed in [8].

Several optimisation approaches are considered for the development of the indirect MPC controller. The optimisation method not be computationally expensive and the cost function should achieve a fast transient response and low total harmonic distortion (THD) of the grid currents. For this thesis, the gradient projection method is chosen as optimisation technique since it is relatively simple to implement, compared to the likes of the interior-point method. The gradient method also converges relatively quickly for the control problem at hand in order to produce an optimal output.

The second objective is to implement the proposed controller on a low-cost FPGA device. The implementation requires careful use of resources, such as multiplier blocks (or DSP slice, which will be mentioned later on) as well as a careful consideration of the timing constraints on the design. A trade-off between time and resource usage must be found to effectively implement the controller in VHDL. The design must be verified and tested by means of a HiL simulation to demonstrate the practical feasibility of the control method.

The results will be compared to the MATLAB equivalent simulations.

The final objective is to test an alternative control strategy developed in [8], known as the small-signal LQR controller. The LQR controller must be implemented in VHDL and tested by means of a HiL simulation, and the results will be compared to the MATLAB based simulation to confirm the functionality of the controller.

1.3 Thesis Summary

This thesis consists of eight chapters, which are summarised below:

Chapter 1: Introduction provides background information on the thesis. The rise in popularity of MPC in the power electronic industry is discussed, after which indirect MPC is explained. The high attenuation property of an *LCL*-filter is mentioned as well as different strategies in damping the resonant frequencies of the *LCL*-filter. A brief overview of other MPC strategies is given, and how they fare compared to the proposed controller. Finally, the thesis objectives are listed, where the main objective is the development and verification of an indirect MPC control strategy implemented on an FPGA device.

Chapter 2: Preliminaries defines and explains concepts and terminology that will be useful to know when reading the chapters that follow. Among the concepts are the level sets of a function and the Hessian matrix.

Chapter 3: Theoretical Background serves as a literature study. The topology of the inverter is given and examined, including the analysis of the switching vectors. The properties of the *LCL*-filter are then examined and the grid is investigated with the voltage and current harmonic requirements. Two modulation schemes are then discussed, namely the well-known carrier-based PWM and space-vector modulation. The following section discusses MPC in detail. A cost function and constraints are then formulated in quadratic programming format. The optimisation approach is then discussed where the gradient projection method is explained. An alternative small-signal LQR control strategy is then derived. The chapter concludes with a summary.

Chapter 4: Controller Design describes the approach in designing the controller for the specific mathematical system model. The control problem is stated and a flow diagram is given depicting the proposed controller.

Chapter 5: Simulation Results is divided into two sections. The first part discusses the simulation results of the proposed controller, where the transients, THD and the grid current spectrum of the system are examined. The robustness of the controller is then tested by decreasing the grid inductance value by half. Different weighting factors and prediction horizon lengths are investigated. The second part discusses the simulation results of the LQR controller with a step in grid current reference.

Chapter 6: FPGA Implementation discusses the design process of both controllers in VHDL. The chapter starts with a few preliminary concepts such as VHDL, an overview of the FPGA for the design, arithmetic in VHDL, delay compensation, and hardware-

in-the-loop simulation. A thorough description of the LQR controller design is given, followed by a detailed design of the proposed controller. The resource usage of both methods is analysed and discussed throughout the chapter.

Chapter 7: HiL Implementation and Results discusses the implementation of a HiL simulation on the FPGA device. Results for both controllers are given and compared to the corresponding MATLAB simulations. Discrepancies between the HiL results and the MATLAB generated results are discussed and recommendations for improvements are made.

Chapter 8: Conclusion and Recommendations is the final chapter. The main findings of the thesis are given and results are discussed. Recommendations for future work and research are proposed.

Chapter 2

Preliminaries

2.1 Introduction

This chapter gives an overview of a few terms and definitions mentioned later on in this thesis. The terms are explained briefly in order to provide clarity.

2.2 Positive Definiteness of a Matrix

A matrix is positive definite if it is a symmetrical matrix and all the eigenvalues of the matrix are positive. The eigenvalues of a matrix are the roots of the polynomial characteristic equation of that matrix. When a function is twice differentiated and the resulting matrix is positive definite, then the function is convex.

2.3 Level Sets of a Function

A level set of a function of two variables $f(x, y)$ forms a curve in a two-dimensional space. The levels set are also referred to as contours. As an example, the level sets of the function

$$f(x, y) = x^2 + y^2 \tag{2.1}$$

form circular curves around the origin, which is shown for $f(x, y) = z = 1, 4, 9$ in Fig 2.1. See Fig 3.11 for another illustration of level sets.

2.4 Conditioning of the Hessian

The matrix $\boldsymbol{H}$ is defined as the Hessian and is a symmetrical matrix of second-order partial derivatives of the scalar-valued cost function J. See [9] for more information regarding the Hessian.

A quadratic function can be written in the form

$$f = \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}\boldsymbol{x} + \boldsymbol{c}^\top \boldsymbol{x} \ , \tag{2.2}$$

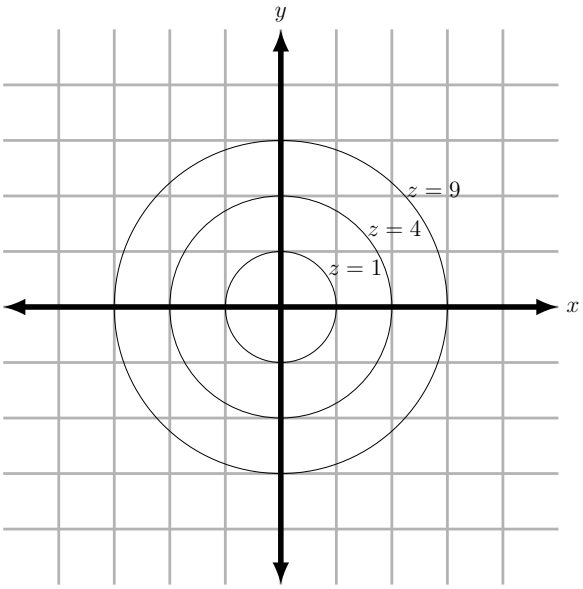


Figure 2.1: Level sets of function $f(x, y)$.

where $\top$ indicates the transpose of a matrix or vector, and $\boldsymbol{x}$ is known as the optimiser, in the case that f is a cost function. The minimum of f can be found by setting the gradient of f equal to zero, i.e.

$$\nabla f = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{c} = 0 \ . \tag{2.3}$$

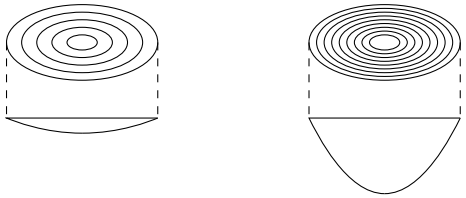
If $\boldsymbol{H}$ is positive semi-definite, the function is still convex, but there exists more than one global minimum. The solution is, therefore not unique. When $\boldsymbol{H}$ is positive definite, there exists a unique solution, which is given as

$$\boldsymbol{x}^* = -\boldsymbol{H}^{-1}\boldsymbol{c} \tag{2.4}$$

where $\boldsymbol{x}^*$ indicates the minimum.

Despite being positive definite, the Hessian can also be ill-conditioned, posing a problem for the optimisation step. The conditioning of a symmetrical matrix is defined as the ratio between its smallest and largest eigenvalue. A badly conditioned $\boldsymbol{H}$ has a conditioning ratio close to infinity. This means that the matrix is close to singular and the inversion thereof becomes computationally intractable. The contours of the level sets of an ill-conditioned $\boldsymbol{H}$ become close to parallel. A representation of the level sets of $\boldsymbol{H}$ is shown in Fig. 2.2 for a small value for λ on the left, as well as a large value on the right hand side. For optimisation techniques such as the gradient method, bad conditioning leads to an especially slow convergence rate. In order to overcome this problem, the weight λ is added to improve the conditioning of $\boldsymbol{H}$. For power electronic converters, this weight is typically placed as a penalty on the control effort. The value of λ cannot, therefore, be increased indefinitely since a large value will cause the converter to become lethargic.

The format in which the optimal solution is determined also effects the conditioning of $\boldsymbol{H}$. When there are multiple vectors that produce the same solution, the optimal solution is, therefore, not unique and adds to the ill-conditioning of $\boldsymbol{H}$. For example, a two-level voltage-source inverter has three vectors that produce the zero-vector component, when



$\lambda_u = 1$ $\lambda_u = 1 \cdot 10^3$

Figure 2.2: Level set contours of function.

the controller is operated in three-phase *abc* format. To improve the conditioning, the controller can be operated in the stationary reference frame $\alpha\beta$, which ensures uniqueness of the vectors.

Chapter 3

Theoretical Background

3.1 Introduction

This chapter provides the reader with key concepts, description of hardware topologies and controller principles that are necessary for the later chapters of this thesis. The second section discusses the topology of a two-level voltage-source inverter in terms of its voltage levels and also explains the stationary reference frame $\alpha\beta$ that is used to reduce the mathematical model of the inverter. The third section explores the properties of an LCL-filter such as attenuation and resonance frequencies. The following section discusses the power grid with the requirements for current and voltage harmonics for when an inverter is connected thereto. The fifth section discusses two modulation schemes, namely normal carrier-based PWM and space vector modulation.

The sixth section provides an introduction to MPC with its advantages and disadvantages, and looks at the difference between direct and indirect MPC. The internal dynamic model is then discussed and constraints for the controller are then explained. The optimisation stage is then explained, and a cost function is derived.

The following section explains the quadratic programming formulation of the cost function and constraints. An optimisation algorithm is discussed in the following section, after which an implementation feasibility study is given.

The ninth section discusses an alternative control technique, known as the small-signal LQR controller. The section starts with an introduction to the LQR controller, after which the steady-state reference signals are calculated. The small-signal model is then derived in discrete-time and the LQR cost function is finally defined.

3.2 Two-level voltage source inverter

3.2.1 Topology of inverter

The two-level voltage source inverter (VSI) in Fig. 3.1 consists of three phase arms. Each phase arm contains two insulated-gate bipolar transistors (IGBTs) semiconductor switches (S_1 to S_6), each with their own freewheeling diode. The supply voltage is denoted by V_d and is connected to two identical capacitors C_d, which form the DC-bus capacitor bank. The connection between the capacitors forms a neutral node n.

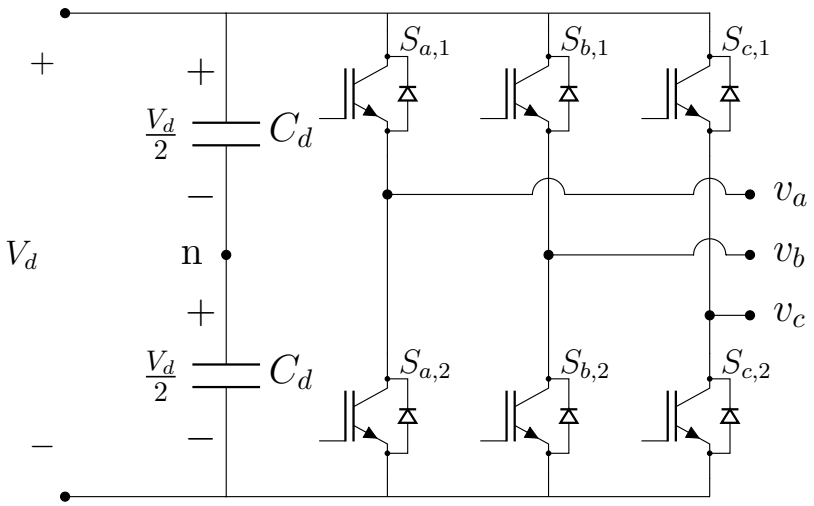


Figure 3.1: Topology of two-level voltage-source inverter.

Each phase arm produces two voltage levels, $\frac{V_d}{2}$ and $-\frac{V_d}{2}$, with respect to the neutral point n. If it is assumed that the neutral point voltage fluctuations are zero and that the voltage across the bus capacitors are equal, then the output voltage of a single phase arm is given by

$$v_x = \frac{V_d}{2} p_x \ , \tag{3.1}$$

where $p_x \in \{-1, 1\}$ denotes the switch position of a certain phase arm and $x \in \{a, b, c\}$ denotes the phase. Table 3.1 shows the relation between the switching states and the phase voltages of a single phase arm. A third state, not indicated in Table 3.1, is known as dead-time, where both switches are switched off to avoid a short circuit during transition.

Table 3.1: Relationship between switching states and phase voltages of a phase arm

p_x	$S_{x,1}$	$S_{x,2}$	v_x
1	1	0	$\frac{V_d}{2}$
-1	0	1	$-\frac{V_d}{2}$

3.2.2 Stationary reference frame and switching vectors

The three-phase system model (discussed in Section 4.2) can be simplified by mapping the *abc* vector to a stationary orthogonal reference frame $\alpha\beta 0$ by using the Clarke transformation matrix

$$\boldsymbol{\xi}_{\alpha\beta 0} = \boldsymbol{K}\boldsymbol{\xi}_{abc} \ . \tag{3.2}$$

These vectors can be mapped back to *abc* using the inverse Clarke transform

$$\boldsymbol{\xi}_{abc} = \boldsymbol{K}^{-1}\boldsymbol{\xi}_{\alpha\beta 0} \ , \tag{3.3}$$

where

$$\boldsymbol{K} = \frac{2}{3}\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \text{ and } \boldsymbol{K}^{-1} = \begin{bmatrix} 1 & 0 & 1 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & 1 \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} & 1 \end{bmatrix} . \tag{3.4}$$

In this thesis, the three-phase system (discussed in Section 4.2), with the star point not connected to the neutral point, is assumed to be balanced and, therefore, the 0-component of the $\alpha\beta 0$ reference frame is omitted [9]. The reduced Clarke transformation matrices are redefined as

$$\boldsymbol{K} = \frac{2}{3}\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \text{ and } \boldsymbol{K}^{-1} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} . \tag{3.5}$$

The two-level inverter has eight possible switching states, which are shown in Table 3.2. The Clarke transform

$$\boldsymbol{p}_{\alpha\beta} = \boldsymbol{K}\boldsymbol{p}_{abc} \tag{3.6}$$

is applied to the abc switching vectors to form eight vectors in $\alpha\beta$. Only seven of these vectors are unique, since the zero vector is produced twice. The vectors are visualised on the $\alpha\beta$ axis in Fig. 3.2. The $\alpha\beta$ output voltage vectors can be calculated with

$$\boldsymbol{v}_{\alpha\beta} = \frac{V_d}{2}\boldsymbol{p}_{\alpha\beta} . \tag{3.7}$$

Table 3.2: Relationship between abc- and $\alpha\beta$ switching vectors.

p_a	p_b	p_c	p_α	p_β
-1	-1	-1	0	0
-1	-1	1	$-\frac{2}{3}$	$-\frac{\sqrt{3}}{2}$
-1	1	-1	$-\frac{2}{3}$	$\frac{\sqrt{3}}{2}$
-1	1	1	$-\frac{4}{3}$	0
1	-1	-1	$\frac{4}{3}$	0
1	-1	1	$\frac{2}{3}$	$-\frac{\sqrt{3}}{2}$
1	1	-1	$\frac{2}{3}$	$\frac{\sqrt{3}}{2}$
1	1	1	0	0

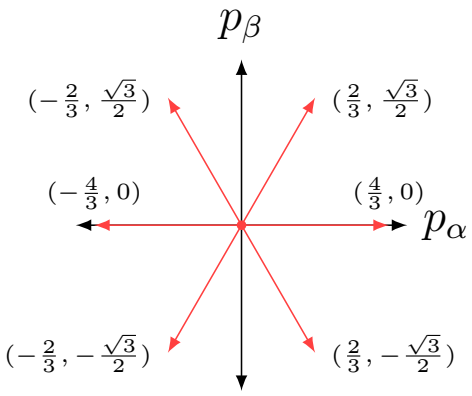


Figure 3.2: Switching vectors in $\alpha\beta$ reference frame.

3.3 LCL-Filter

Grid-connected inverters are often accompanied by *LCL*-filters[1] to reduce the output current ripple of the inverter and grid currents. The per-phase topology in Fig. 3.3 shows the filter inductor L_f, the grid inductance L_g and the filter capacitor C.

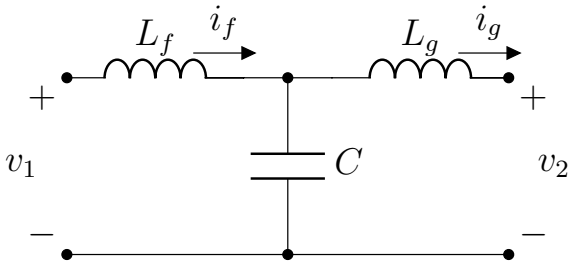


Figure 3.3: Per-phase *LCL*-filter topology.

The *LCL*-filter is a third -order filter that introduces two resonant frequencies to the plant. To investigate the frequency response of the grid current as well as the inverter current of the *LCL*-filter, consider the two transfer functions

$$H_1(s) = \frac{I_f(s)}{V_1(s)} = \frac{1 + L_g C s^2}{L_f L_g C s^3 + (L_f + L_g)s} \tag{3.8}$$

and

$$H_2(s) = \frac{I_g(s)}{V_1(s)} = \frac{1}{L_f L_g C s^3 + (L_f + L_g)s}\ , \tag{3.9}$$

where the former is defined as the transfer function of the inverter voltage to the inverter current and the latter is defined as the transfer function of the inverter voltage to the grid current of the *LCL*-filter. For the grid current transfer function, the grid frequency is assumed to be 50 Hz and acts as a short-circuit for all other frequencies. The frequency magnitude response of the transfer functions is shown in Fig. 3.4. The filter parameters chosen for the bode plot are given in Table 3.3.

[1]To be more specific, an *LC*-filter is connected to the grid. The latter is modelled as an inductor (forming the second *L*) along with a voltage source. Throughout the text, the filter is referred to as *LCL* to emphasize the inclusion of the grid inductance.

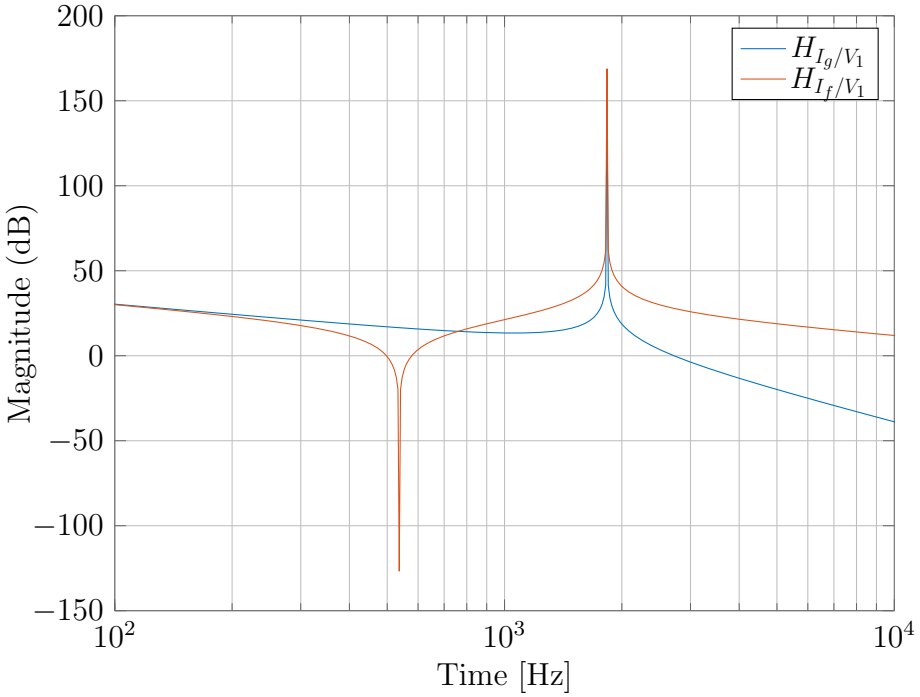


Figure 3.4: Magnitude frequency plot of the *LCL*-filter.

From the transfer functions (3.8) and (3.9) and the bode plot, it can easily be seen how the *LCL*-filter attenuates the higher frequencies, albeit at the cost of two resonance frequencies.

Table 3.3: *LCL*-filter parameters for frequency response.

Parameter	Symbol	Value
Filter inductance	L_f	4.17 µH
Grid inductance	L_g	44.38 µH
Filter capacitance	C	1.98 mF

The two resonant frequencies in Fig. 3.4 are $f_{res,1} = 1831.98\,\text{Hz}$ and $f_{res,2} = 536.9\,\text{Hz}$. These frequencies can be calculated with

$$f_{res,1} = \frac{1}{2\pi}\sqrt{\frac{L_f + L_g}{L_f C L_g}} \tag{3.10}$$

and

$$f_{res,2} = \frac{1}{2\pi}\frac{1}{\sqrt{L_f C}}\,. \tag{3.11}$$

There are two approaches in damping the resonant frequencies, namely passive and active damping [3]. The passive damping approach involves adding a physical resistor to the *LCL* network, which decreases the system efficiency. The active damping approach is, therefore, more attractive and there are three strategies that form the basis of active damping in practise. Multiloop control consists of outer control loops to regulate grid current and inner control loops that stabilise the system [10, 11]. Virtual resistors can

also be added to the filter to actively dampen the resonant frequencies. When deriving the mathematical model of the filter, a theoretical resistor is added to the circuit to act as a virtual resistance. A virtual resistor can either be added in series or parallel to the filter inductor, capacitor or both. Depending on the configuration, an additional current sensor and differentiator might be necessary (in case of the series resistance) or an additional voltage amplifier [12, 13]. Filter-based strategies use genetic algorithms to generate optimal parameters for an active damping filter [13, 14].
To summarize, the high attenuation property of the *LCL*-filter makes it more attractive than the series *L*-filter when it comes to grid applications [3]. However, the high attenuation does not come without some drawbacks as the filter requires damping for the resonant frequency(ies) that it introduces.

3.4 Grid

A representation of a grid-tied inverter is shown in Fig. 3.5. The inverter output voltages v_{an}, v_{bn} and v_{cn} are connected to the point of common coupling (PCC) via a transformer. Since there are usually no accurate models of the grid, it is represented by an inductance L_g in series with a resistance R_g as well as a three-phase grid voltage $\boldsymbol{v}_g$ [9].

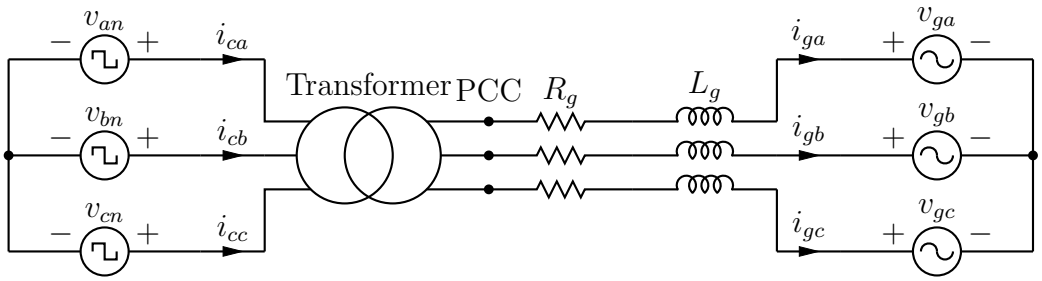


Figure 3.5: Grid-connected inverter representation (reproduced from [9]).

The strength of a grid can be measured by the short-circuit ratio

$$k_{sc} = S_{sc}/S_c \ , \tag{3.12}$$

where S_{sc} denotes the short-circuit power of the grid and S_c denotes the rated power of the inverter. The short-circuit power is defined as the power delivered to the PCC in case of a three-phase fault and can be expressed as

$$\begin{aligned} S_{sc} &= 3(\frac{V_g}{\sqrt{3}})^2/|Z_g| \\ &= V_g^2/|Z_g| \end{aligned} \tag{3.13}$$

where V_g is the grid line-to-line voltage (in rms) and $Z_g = j\omega_g L_g + R_g$ is the grid impedance, with

$$|Z_g| = \sqrt{(\omega_g L_g)^2 + R_g^2} \ . \tag{3.14}$$

A higher ratio for k_{sc} indicates a strong (also called stiff) grid (typically $k_{sc} > 20$), where the power of the grid dominates the power of the inverter. When k_{sc} is lower than 8,

it is a sign of a weak grid, since the grid impedance is much larger than the impedance of the inverter [9]. In the case of a weak grid, care must be taken with the harmonics injected by the inverter into the grid at the PCC, since the voltage is very sensitive to any variation in the load. See [9] for more information on strong and weak grids.
The design of a grid-tied inverter requires close attention to the grid standards that are imposed at the PCC. These standards place constraints on the magnitude of specific voltage and current harmonics. The IEEE 519 standard for current harmonics and the IEC 61000-2-4 standard for voltage harmonics are commonly imposed standards [15, 16]. In South Africa, grid specifications are given by the NRS-048 user specification part two and four as issued by the Technical Governance Department (Eskom) on behalf of the NRS User Group [1]. The voltage harmonic compatibility levels are shown in Table 3.4.

Table 3.4: Voltage harmonics compatibility levels (reproduced from [1]).

Odd harmonics				**Even harmonics**	
Not multiples of 3		**Multiples of 3**			
Order (h)	Magnitude (%)	Order (h)	Magnitude (%)	Order (h)	Magnitude (%)
5	6	3	5	2	2
7	5	9	1.5	4	1
11	3.5	15	0.5	6	0.5
13	3	21	0.3	8	0.5
$17 \leq h \leq 49$	$\{2.27 \times (\frac{17}{h})\} - 0.27$	$21 \leq h \leq 45$	0.2	$10 \leq h \leq 50$	$\{0.25 \times (\frac{10}{h})\} + 0.25$

NRS-048 specifies the current harmonic limits as a function of the above-mentioned short-circuit ratio k_{sc}. The standard for the limit of current injection levels is given in Table 3.5.

Table 3.5: Maximum current injection levels (reproduced from [1]).

Maximum current harmonic distortion h (% of I_L)						
$\boldsymbol{I_r = I_{sc}/I_L}$	$\boldsymbol{h < 11}$	$\boldsymbol{11 \leq h < 17}$	$\boldsymbol{17 \leq h < 23}$	$\boldsymbol{23 \leq h < 35}$	$\boldsymbol{h > 35}$	**TDD**
$I_r < 20$	4	2	1.5	0.6	0.3	5
$20 \leq I_r < 50$	7	3.5	2.5	1	0.5	8
$50 \leq I_r < 100$	10	4.5	4	1.5	0.7	12
$100 \leq I_r < 1000$	12	5.5	5	2	1	15
$I_r > 1000$	15	7	6	2.5	1.4	20

It is typically more difficult to adhere to grid voltage standards than with current harmonic standards, since the magnitude of a voltage harmonic at the PCC is determined by the amplitude of the injected voltage harmonic of the inverter as well as the ratio of the transformer impedance to the grid impedance. These impedances form a voltage divider that is independent of frequency and, therefore, the voltage harmonics are not attenuated at higher frequencies. The current harmonics are, however, attenuated at higher frequencies, since the impedance increases linearly with the frequency [9]. For this reason, an *LCL*-filter is used, as the additional capacitor helps attenuate some of the voltage harmonics.
In summary, when connecting an inverter to a grid, it is important to take into account the strength/stiffness of the specific grid system, since a weaker grid is more sensitive to any variation in the load.

3.5 Modulation technique

3.5.1 Carrier-based pulse-width modulation

When it comes to power electronic converters, the most common strategy of driving the semiconductor switches is known as pulse-width modulation (PWM). Carrier-based PWM involves the comparison of a real-valued reference signal u_{ref} with frequency f_1, to a carrier signal with frequency f_c. For a two-level inverter, the amplitude of the carrier signal ranges from -1 to 1 and the semiconductor switch is turned on when the amplitude of the reference signal is larger than that of the carrier signal. See Fig. 3.6 for an illustration of this principle. The output of the pulse-width modulator, therefore, directly translates to the inverter switch positions. A high ratio is usually chosen between the carrier and fundamental frequencies, i.e $f_c >> f_1$, in order to establish a low average frequency of the voltage or current output [17].

A problem that arises with PWM is the harmonic content that is produced at the output in addition to the fundamental component u_{ref}, which is known as the carrier multiple-, baseband-, and sideband harmonics. The locations of these harmonics can be described by the equation

$$f_{\mu\nu} = \mu f_c + \nu f_1 \tag{3.15}$$

with $\mu \in \mathbb{N}$ and $\nu \in \mathbb{Z}$, where μ and ν indicate integer multiples of the carrier and fundamental frequencies, respectively. See [17] for a comprehensive study on PWM harmonics. In order to assess the effectiveness of a modulation method, a metric standard is necessary. One such method is known as the total harmonic distortion (THD), which is defined as the ratio between the unwanted components at the output and the amplitude of the fundamental component. The current THD is defined as

$$I_{THD} = \frac{1}{I_1}\sqrt{\sum_{n\neq 1}(\bar{i}_n)^2}\ , \tag{3.16}$$

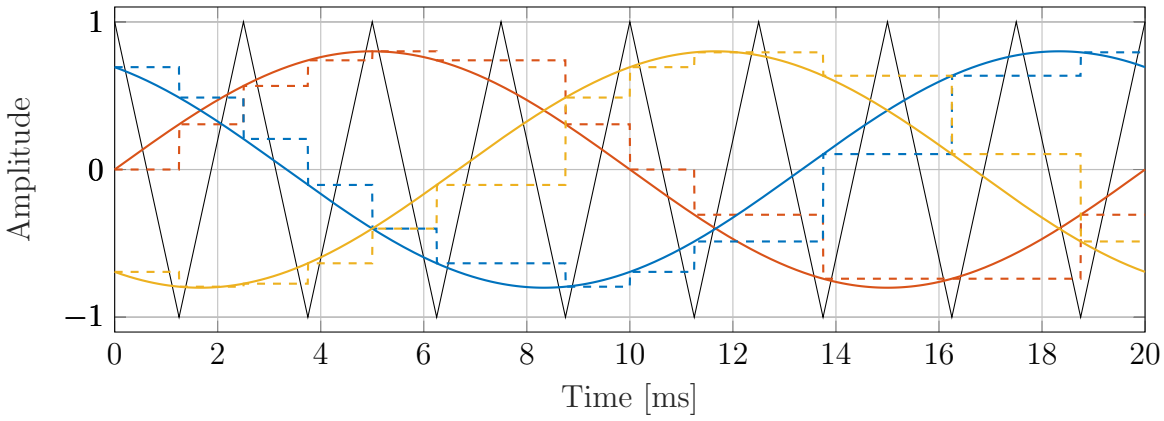
where I_1 is the amplitude of the fundamental current component and $\bar{i}_n$ is the nth current harmonic amplitude, occurring at nf_1 for $n \in \{2, 3, \ldots\}$. It is important to note that this definition is only valid for single-phase currents. In order to calculate the THD of three-phase currents, the THD of each phase must be found separately after which the total THD can be calculated by taking the mean value of the phases [9].

With carrier-based PWM (CB-PWM), the reference signal can either be sampled naturally or regularly. Natural sampling refers to the method in which the switches are switched at the instant when the amplitudes of the reference signal crosses the carrier. Modern controllers are, however, implemented digitally and require discrete-valued signals that are sampled regularly. There are two types of regular sampling. The first is know as *symmetric regular sampling*, where the reference signal is sampled once in a carrier period $T_c = \frac{1}{f_c}$. The second is called *asymmetric regular sampling*, where the reference signal is sampled twice in one carrier period: at the upper and lower peaks of the carrier. For both of these sampling techniques, the reference is held constant until the next sample. Asymmetric regular sampled CB-PWM is illustrated in Fig. 3.6.

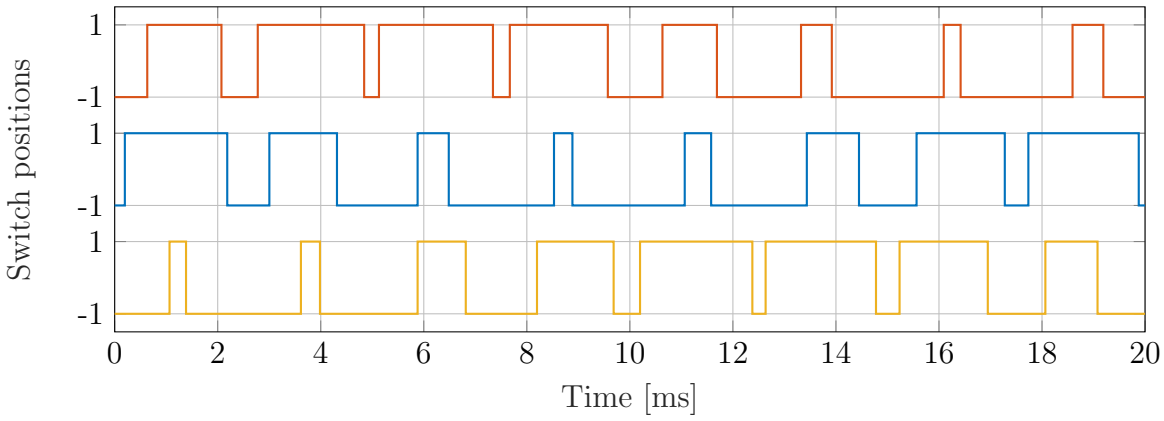
The reference signals for a three-phase two-level VSI (in *abc* format) are defined as

$$\boldsymbol{u}_{ref}(t) = m \begin{bmatrix} \sin(\omega_1 t) \\ \sin(\omega_1 t - \frac{2\pi}{3}) \\ \sin(\omega_1 t - \frac{4\pi}{3}) \end{bmatrix} , \tag{3.17}$$

where m is known as the modulation index and ranges between 0 and 1 for the linear modulation region. The b and c phases lag the a-phase by 120° and 240°, respectively. The three-phase reference signals are compared to the same carrier signal $c(t)$.



(a) Illustration of reference $\boldsymbol{u}_{abc}(t)$ and carrier $c(t)$ signals of CB-PWM. The solid colour lines are the reference signals and the solid black line is the carrier. The dashed colour lines represent the asymmetrically sampled reference signals.



(b) Illustration of the three-phase gating signals $\boldsymbol{p}_{abc}(t)$ for the inverter, starting with phase a at the top.

Figure 3.6: Depiction of asymmetrically regularly sampled CB-PWM, with $f_1 = 50\,\text{Hz}$, $f_c = 400\,\text{Hz}$, and $m = 0.8$.

3.5.2 Space-vector modulation

An attractive alternative for CB-PWM is space-vector modulation (SVM), which provides better performance in terms of reduced current harmonics and fully utilizing the bus DC-link voltage [18, 19]. When examining the amplitudes of the three-phase modulating signals in Fig. 3.6a, the line-to-line difference between two phases at any given moment is $\sqrt{3}m$, whereas the difference in peak amplitudes is $2m$. This indicates that the bus voltage is not fully utilized [9]. For two-level inverters, it is well known that with the

addition of a common offset voltage term, the line-to-line voltage can be boosted to $m = \frac{2}{\sqrt{3}} \approx 1.15$, thus fully exploiting the bus voltage [20]. The common mode term is given by

$$\overline{u}_0(t) = -\frac{\max(\boldsymbol{u}_{abc}(t)) + \min(\boldsymbol{u}_{abc}(t))}{2}\,, \tag{3.18}$$

which centres the active space-vectors during the switching period, therefore achieving SVM equivalence. Fig. 3.7 shows the updated SVM equivalent three-phase reference signals $\overline{\boldsymbol{u}}_{abc}(t) = \boldsymbol{u}_{abc}(t) + \overline{u}_0(t)$.

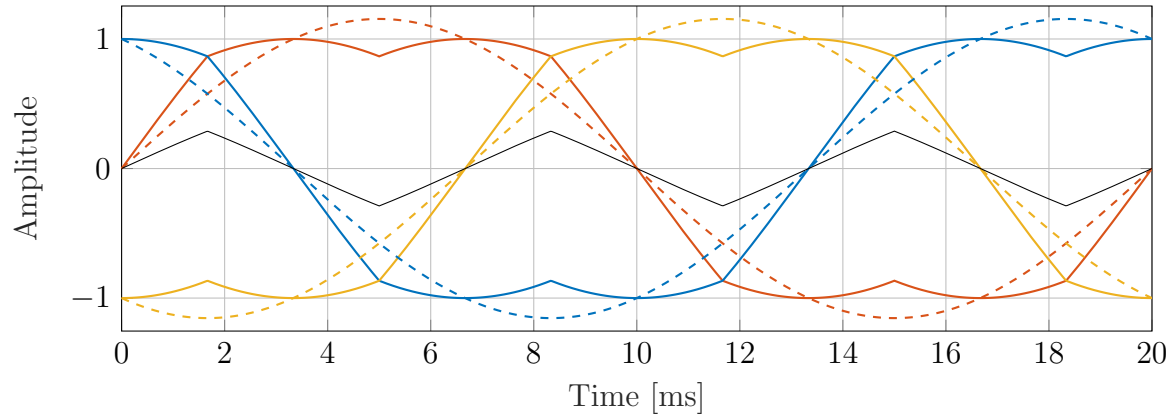
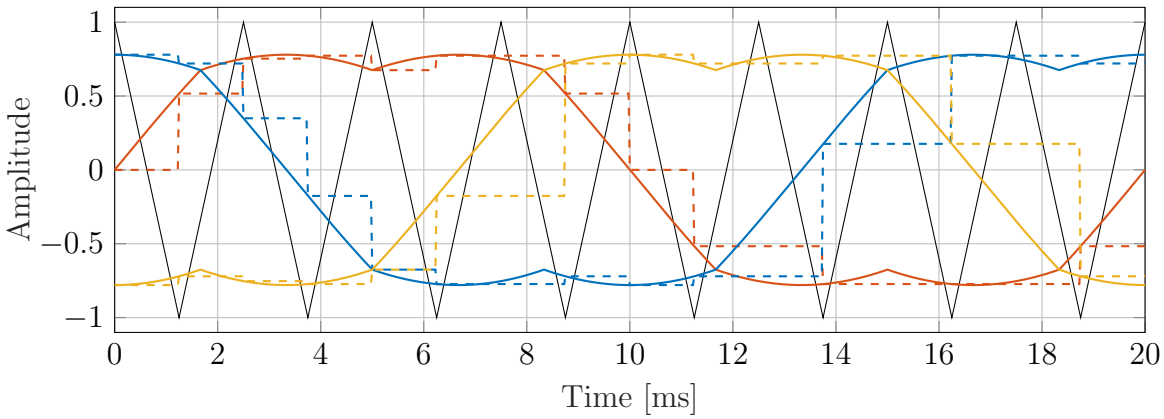
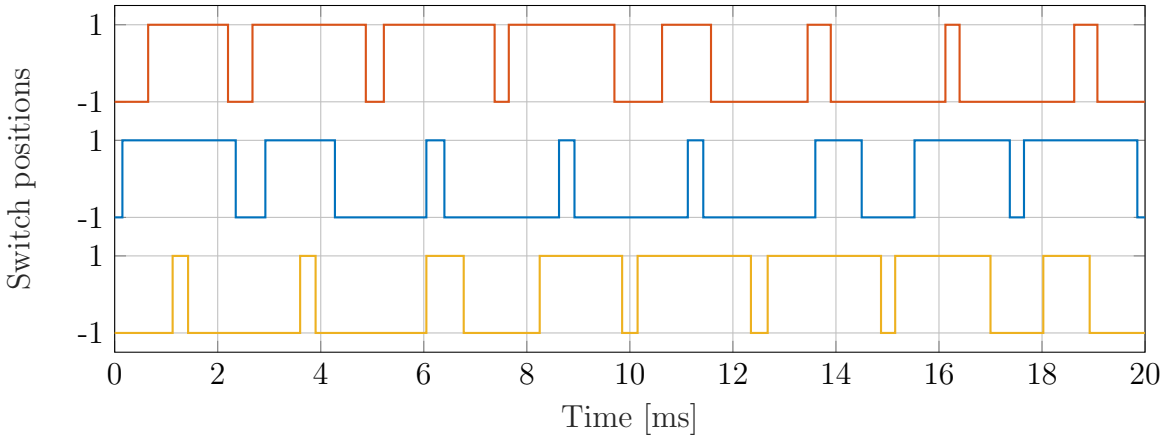


Figure 3.7: Illustration of SVM equivalent CB-PWM. The solid colour lines are the new reference signals $\overline{\boldsymbol{u}}_{abc}(t)$, the dashed colour lines are the normal references $\boldsymbol{u}_{abc}(t)$ with $m = \frac{2}{\sqrt{3}}$, $f_1 = 50\,\text{Hz}$ and $f_c = 400\,\text{Hz}$. The common mode term $\overline{u}_0$ is shown as a solid black line.



(a) Illustration of reference $\overline{\boldsymbol{u}}_{abc}(t)$ (solid coloured lines) and carrier $c(t)$ (black) signals of SVM equivalent PWM. The dashed colour lines represent the asymmetrically sampled reference signals.



(b) Switch gating signals $\boldsymbol{p}_{abc}(t)$ equivalent to SVM.

Figure 3.8: Illustration of SVM equivalent CB-PWM with the addition of a common voltage term $\overline{u}_0$.

3.6 Model predictive control

3.6.1 Introduction

Traditionally, model predictive control (MPC) has been limited to the process industry, where the plant dynamics tend to be slow. This was due to the limited computational power needed to solve the underlying optimisation problem in real time. However, with the turn of the century came a rise in processing power, e.g. in the form of field programmable gate arrays (FPGA) and more powerful microcontrollers, expanding the use of MPC to the power electronics industry. Due to the non-linear switching nature of power electronic systems, the controller is more difficult to design, analyse and verify. [9].

MPC is implemented by using an internal dynamic model of a system, as well as model-specific constraints. This model is used to predict the system behaviour over a prediction horizon based on the current sampled values of the state variables. The predicted behaviour is then compared to a reference value within a cost function (forming an error value), which can also contain additional penalties, e.g. on control effort. An optimisation stage then minimises the cost function subjected to constraints, providing an optimal sequence of converter inputs over a prediction horizon to the system. A *receding horizon* principle is then applied, where only the first input of the sequence is used as input to the system, forming a feedback loop. At the next sampling instant, the whole process is repeated with updated measurements. The method provides stability and ensures a certain measure of robustness.

Direct vs indirect MPC

Within the context of power electronic converters, MPC can be divided into two categories. The first is known as direct MPC (also referred to as finite-control set MPC), where the manipulated variable directly corresponds to the inverter switch positions. While the internal dynamic model of the system remains linear, the controller output consists out of integer variables corresponding to the switching states. Since the constraints are non-

convex, typical convex optimisation techniques cannot be applied to direct MPC. One possible optimisation method, known as sphere decoding, has been experimentally verified in [21]. Sphere decoding attempts to search through a lattice of feasible points which lie within a sphere with a certain radius. The search space can be reduced by decreasing the radius of the sphere, thereby also reducing the required number of computations [22].

The second type of MPC is known as indirect MPC, where a modulator is placed between the controller and the plant. The manipulated variable is now real-valued and typically used as a voltage reference to the pulse-width modulator. The main advantage of indirect MPC is that the control problem is convex, which can be solved efficiently with the existing methods for convex optimisation, given that the cost function is quadratic (see [23]).

The switching behaviour of the system can be linearised through an averaging technique, such as the one described in [24], where the pulse-width modulator is modelled as a gain. Averaging, however, only conceals the switching behaviour of power electronic systems at higher frequencies and should be avoided when working with a low switching frequency, in which case the controller and modulation scheme should take into account the non-linear nature of the plant [9].

Another method is presented in [6], where the pulse-width modulator is modelled as a piecewise-affine (PWA) function, which results in a PWA model of the entire system. The control problem is solved off-line and the control laws are stored in lookup tables, which results in a state-space that is partitioned into a set of polyhedra, where each polyhedron is associated with a PWM control law. From the lookup table, the optimal manipulated variable can be read in real time. This approach is known as explicit MPC, and is limited to problems with smaller horizons and a low-dimensional state vector, and is therefore not considered in this document. For more information on explicit MPC, see [25].

Advantages and drawbacks of MPC

When compared to other classical control methods, such as proportional-integral-derivative (PID) controllers, there are a number of advantages to MPC that stands out. A few of the principle advantages are:

- MPC can intuitively be applied to both linear and non-linear systems, where the control problem is formulated in the time domain. The switching nature of power electronics is addressed by MPC by including the internal dynamic model in the control problem formulation, thus not requiring an averaged model or linearisation.

- Constraints can easily be imposed on controller outputs, inputs, and states within MPC. Multiple control objectives can be added to the control problem formulation, with weights applying a penalty to each control objective, thus providing the user with multiple control modes in a single controller. In contrast to MPC, proportional-integral (PI) controllers require an additional anti-windup mechanism to prevent integrator saturation [9].

- Classical controllers such as PID are often limited to single-input single-output (SISO) systems. In order to tackle multi-input multi-output (MIMO) systems,

multiple decoupled PID control loops would be required, where each must be tuned individually. This is, however, a daunting task since the controllers interact with each other unfavourably in practise. A MIMO MPC control problem can normally be formulated with ease.

There are, however, some challenges for MPC. One drawback, in the case of direct MPC, is the exponential increase in computational complexity when it comes to the optimisation stage with an increase in the prediction horizon. The formulation of the control problem is simple when compared to the optimisation technique. A suitable method must be chosen carefully, as implementation on a controller, such as an FPGA, can prove very challenging.

3.6.2 Internal dynamic model

MPC uses an internal dynamic model of the system that it aims to control. A block diagram representation of an indirect model predictive controller is shown in Fig. 3.9. The output variable $\boldsymbol{y}$ is subtracted from the reference $\boldsymbol{y}^*$ and given to the controller. The controller calculates the optimal PWM reference signal $\boldsymbol{u}$, which is then given to a pulse-width modulator. The plant receives the switching signal $\boldsymbol{p}$. This section describes the plant model.

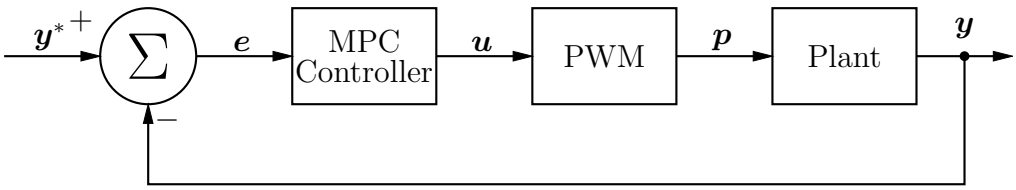


Figure 3.9: Block diagram depiction of an indirect control system.

The state vector is defined as $\boldsymbol{x}(t) \in \mathbb{R}^{n_x}$, the input vector as $\boldsymbol{p}(t) \in \mathbb{R}^{n_u}$, and the output vector as $\boldsymbol{y}(t) \in \mathbb{R}^{n_y}$, where n_x, n_u, and n_y denotes the dimension of each respective vector. The continuous-time state-space model of the plant can be expressed by a system of linear differential equations

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{F}\boldsymbol{x}(t) + \boldsymbol{G}\boldsymbol{p}(t) + \boldsymbol{P}\boldsymbol{v}_g(t) \tag{3.19}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) \ , \tag{3.20}$$

where $\boldsymbol{F} \in \mathbb{R}^{n_x \times n_x}$ is the system matrix, $\boldsymbol{G} \in \mathbb{R}^{n_x \times n_u}$ is the input matrix, and $\boldsymbol{C} \in \mathbb{R}^{n_y \times n_x}$ is the output matrix. The third term in (3.19) includes the effect of the grid voltage, with $\boldsymbol{P} \in \mathbb{R}^{n_x \times n_v}$ and $\boldsymbol{v}_g(t) \in \mathbb{R}^{n_v}$ denoting the grid voltage matrix and reference vector, respectively.

In order to translate the continuous-time model to the discrete-time domain, a constant sampling interval $T = T_c/2$ must be employed[2]. Since the pulse-width modulator output $\boldsymbol{p}(t)$ is non-linear, the discrete equivalent of the continuous-time linear model is also non-linear. This problem can be overcome by approximating the discrete value $\boldsymbol{p}(k)$ of $\boldsymbol{p}(t)$

[2]The sampling interval is half of the carrier period because of asymmetric regular sampling.

as a zero-order-hold (ZOH) equivalent, effectively ignoring the PWM switching. The discrete-time equivalent of (3.19) and (3.20) can be written as

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{p}(k) + \boldsymbol{\mathcal{V}}\boldsymbol{v}_g(k) \tag{3.21}$$
$$\boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k)\ , \tag{3.22}$$

where $k \in \mathbb{N} = \{0, 1, 2 \ldots\}$ indicates the current sampling instant, and the matrices $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{\mathcal{V}}$ can be calculated as

$$\boldsymbol{A} = \boldsymbol{e}^{\boldsymbol{F}T} \tag{3.23}$$
$$\boldsymbol{B} = -\boldsymbol{F}^{-1}(\boldsymbol{I}_{n_x} - \boldsymbol{A})\boldsymbol{G} \tag{3.24}$$
$$\boldsymbol{\mathcal{V}} = -\boldsymbol{F}^{-1}(\boldsymbol{I}_{n_x} - \boldsymbol{A})\boldsymbol{P}\ . \tag{3.25}$$

The symbol $\boldsymbol{e}$ denotes the matrix exponential and $\boldsymbol{I}_{n_x}$ is the identity matrix with an appropriate dimension. In the case where the matrix exponential proves computationally intractable, $\boldsymbol{A}$ can be calculated using the forward Euler approximation

$$\boldsymbol{A} = \boldsymbol{I}_{\boldsymbol{n_x}} + \boldsymbol{F}T + \frac{(\boldsymbol{F}T)^2}{2!} + \frac{(\boldsymbol{F}T)^3}{3!} \ldots \tag{3.26}$$

with an appropriate amount of terms (three will typically suffice if the sampling interval is short enough). If we define $\boldsymbol{\Psi} = \boldsymbol{I}_{\boldsymbol{n_x}} + \frac{\boldsymbol{F}T_s}{2!} + \frac{(\boldsymbol{F}T)^2}{3!} + \ldots$, then matrices $\boldsymbol{B}$ and $\boldsymbol{\mathcal{V}}$ can be derived using

$$\boldsymbol{B} = \int_0^T \boldsymbol{e}^{\boldsymbol{F}\theta} d\theta \boldsymbol{G} = \boldsymbol{\Psi}\boldsymbol{G}T \tag{3.27}$$
$$\therefore\ \boldsymbol{\mathcal{V}} = \boldsymbol{\Psi}\boldsymbol{P}T\ . \tag{3.28}$$

3.6.3 Constraints

During the optimisation stage, constraints can be placed on the state-, input-, and output variables of the form

$$\boldsymbol{x}(k) \in \boldsymbol{\mathcal{X}} \subseteq \mathbb{R}^{n_x} \tag{3.29}$$
$$\boldsymbol{u}(k) \in \boldsymbol{\mathcal{U}} \subseteq \mathbb{R}^{n_u} \tag{3.30}$$
$$\boldsymbol{y}(k) \in \boldsymbol{\mathcal{Y}} \subseteq \mathbb{R}^{n_y}\ . \tag{3.31}$$

There are two types of constraints, namely *soft* and *hard*. In direct MPC, hard constraints are placed on the outputs of the control loop $\boldsymbol{u}(k)$ since the manipulated variable corresponds directly to the position of a switch. These constraints cannot, therefore be violated. A soft constraint is typically applied in a system where it can be slightly violated, e.g. as an upper bound on a current reference to avoid current surges [9]. An advantage of soft constraints is that the control problem cannot become infeasible when compared to hard constraints.

In the case of indirect MPC, where a pulse-width modulator is present, a hard constraint is placed on each of the three-phase PWM reference signals. The references are, therefore, continuously bounded to the amplitude of the PWM carrier signal, i.e.

$$\boldsymbol{\mathcal{U}} = [-1, 1]^{n_u}\ . \tag{3.32}$$

3.6.4 Optimisation stage

Cost function formulation

The control problem consists of formulating a desirable cost function that addresses specific control objectives. The cost function uses future state values, outputs, and manipulated variables and generates a single scalar value. The optimal sequence of inputs can be obtained by minimising this scalar value with constraints by means of an optimisation technique. A general cost function can be defined as

$$J(\boldsymbol{x}(k), \boldsymbol{x}^*(k), \boldsymbol{U}(k)) = \sum_{\ell=k}^{k+N_p-1} \Lambda(\boldsymbol{x}(\ell), \boldsymbol{u}(\ell)) \ , \tag{3.33}$$

where Λ is a function containing weights, and is summed over a prediction horizon N_p to produce the scalar value J. The cost function penalizes the predicted system behaviour by comparing the future values of $\boldsymbol{x}$ to its reference $\boldsymbol{x}^*$, thus producing an error value. With direct MPC, a penalty would also be placed on the switching of the inverter, whereas with indirect MPC, a penalty is placed on the *change* in PWM reference signal $\boldsymbol{u}$ (in both cases the control effort is penalised). The inputs to the cost function are the current state vector $\boldsymbol{x}(k)$ and the sequence of manipulated variables

$$\boldsymbol{U}(k) = [\boldsymbol{u}^\top(k)\ \boldsymbol{u}^\top(k+1) \dots \boldsymbol{u}^\top(k+N_p-1)]^\top \ , \tag{3.34}$$

where $\boldsymbol{U}(k) \in \boldsymbol{\mathcal{U}}^{N_p}$ and T indicates the transpose of a vector or matrix. The tracking error of the state vector as well as the change in PWM reference signal can be incorporated into a quadratic formulation of the cost function above. Throughout this document, the output vector will always be equal to the input, i.e. $\boldsymbol{x}(k) = \boldsymbol{y}(k)$, resulting in the output matrix $\boldsymbol{C} = \boldsymbol{I}_{n_y}$. Therefore, from here on forward, the output state equation (3.22) will be omitted. The cost function can now be written as

$$J(\boldsymbol{x}(k), \boldsymbol{x}^*(k), \boldsymbol{U}(k)) = \sum_{\ell=k}^{k+N_p-1} ||\boldsymbol{x}^*(\ell+1) - \boldsymbol{x}(\ell+1)||^2_{\boldsymbol{Q}} + \lambda_u ||\Delta \boldsymbol{u}(\ell)||^2_2 \ , \tag{3.35}$$

with $\boldsymbol{Q}$ being a matrix with dimensions $n_x \times n_x$, which can be used to put weights on specific state variables. As an example, for a state vector consisting of grid- and inverter currents as well as filter capacitor voltages (in the case of a grid-tied inverter via *LCL*-filter), a larger weight can be placed on the grid currents to prioritize the controller effort thereon. A weight λ_u is also placed on the change in PWM reference signal, which is defined as $\Delta \boldsymbol{u}(\ell) = \boldsymbol{u}(\ell) - \boldsymbol{u}(\ell-1)$. The squared 2-norm, or Euclidian norm, is applied to each control objective, resulting in a quadratic cost function.

Since the controller and plant are separated by a modulator, the switching frequency is fixed. The purpose of λ_u is to increase the robustness of the controller to external noise. With a higher value for λ_u the controller reacts more slowly to disturbances, prioritizing the change in the PWM reference. However, as λ_u tends to zero, the controller becomes more sensitive, prioritizing the tracking error, which decreases the harmonic distortions. In order to write the cost function more compactly in a vector format, (3.21) is firstly applied to itself recursively over N_p [26], which results in the state trajectory vector

$$\begin{aligned}\boldsymbol{X}(k) &= [\boldsymbol{x}^\top(k+1)\ \boldsymbol{x}^\top(k+2)\ \dots\ \boldsymbol{x}^\top(k+N_p)]^\top \\ &= \boldsymbol{\Gamma}\boldsymbol{x}(k) + \boldsymbol{\Upsilon}\boldsymbol{U}(k) + \boldsymbol{\Omega}\boldsymbol{V}_g(k)\ ,\end{aligned} \tag{3.36}$$

where

$$\boldsymbol{\Gamma} = \begin{bmatrix}(\boldsymbol{A})^\top & (\boldsymbol{A}^2)^\top & \cdots & (\boldsymbol{A}^{N_p})^\top\end{bmatrix}^\top\ , \tag{3.37}$$

$$\boldsymbol{\Upsilon} = \begin{bmatrix}\boldsymbol{B} & 0 & \cdots & 0 \\ \boldsymbol{AB} & \boldsymbol{B} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ \boldsymbol{A}^{N_p-1}\boldsymbol{B} & \boldsymbol{A}^{N_p-2}\boldsymbol{B} & \cdots & \boldsymbol{B}\end{bmatrix}\ , \boldsymbol{\Omega} = \begin{bmatrix}\boldsymbol{\mathcal{V}} & 0 & \cdots & 0 \\ \boldsymbol{A\mathcal{V}} & \boldsymbol{\mathcal{V}} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ \boldsymbol{A}^{N_p-1}\boldsymbol{\mathcal{V}} & \boldsymbol{A}^{N_p-2}\boldsymbol{\mathcal{V}} & \cdots & \boldsymbol{\mathcal{V}}\end{bmatrix} \tag{3.38}$$

$$\text{and } \boldsymbol{V}_g(k) = [\boldsymbol{v}_g^\top(k+1)\ \boldsymbol{v}_g^\top(k+2)\ \cdots\ \boldsymbol{v}_g^\top(k+N_p)]^\top\ . \tag{3.39}$$

The weight matrix $\boldsymbol{Q} = \text{diag}(k_1, k_2, \dots, k_{n_x})$ is expanded as $\boldsymbol{Q}_c = \text{diag}(\boldsymbol{Q}, \boldsymbol{Q}, \dots, \boldsymbol{Q}) \in \mathbb{R}^{N_p n_x \times N_p n_x}$ and the cost function is reformulated as [9]

$$J(\boldsymbol{x}(k), \boldsymbol{U}(k)) = ||\boldsymbol{X}(k) - \boldsymbol{X}^*(k)||^2_{\boldsymbol{Q}_c} + \lambda_u ||\boldsymbol{S}\boldsymbol{U}(k) - \boldsymbol{E}\boldsymbol{u}(k-1)||^2_2\ , \tag{3.40}$$

with $||\boldsymbol{\zeta}||^2_{\boldsymbol{Q}_c} = \boldsymbol{\zeta}^\top \boldsymbol{Q}_c \boldsymbol{\zeta}$. The auxiliary matrices are defined as

$$\boldsymbol{S} = \begin{bmatrix}\boldsymbol{I}_3 & \boldsymbol{0}_{3\times3} & \cdots & \boldsymbol{0}_{3\times3} \\ -\boldsymbol{I}_3 & \boldsymbol{I}_3 & \cdots & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & -\boldsymbol{I}_3 & \cdots & \boldsymbol{0}_{3\times3} \\ \vdots & \vdots & & \vdots \\ \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \vdots & \boldsymbol{I}_3\end{bmatrix} \text{ and } \boldsymbol{E} = \begin{bmatrix}\boldsymbol{I}_3 \\ \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} \\ \vdots \\ \boldsymbol{0}_{3\times3}\end{bmatrix}\ . \tag{3.41}$$

The derivation of (3.40) can be found in Appendix C.

Optimisation problem

The optimal sequence of PWM references $\boldsymbol{U}_{opt}(k)$ can be calculated by minimising the cost function in (3.40) subject to the constraints and internal dynamic model of the plant over a prediction horizon. The optimization problem is defined as

$$\begin{aligned}\boldsymbol{U}_{opt}(k) = \arg\underset{\boldsymbol{U}(k)}{\text{minimise}}\ & J(\boldsymbol{x}(k), \boldsymbol{x}^*(k), \boldsymbol{U}(k)) \\ \text{s.t. } & \boldsymbol{U}(k) \in \mathbb{U}\ ,\end{aligned} \tag{3.42}$$

with $\mathbb{U} = \boldsymbol{\mathcal{U}}^{N_p}$ representing the feasible set of all possible PWM reference trajectories for $\boldsymbol{U}(k)$.

3.7 Receding horizon principle

Consider the block diagram depiction of the receding horizon principle Fig. 3.10. At the current time instant k, the optimisation problem computes the optimal sequence of PWM reference signals $\boldsymbol{U}_{opt}(k)$. To close the feedback loop, only the first element of $\boldsymbol{U}_{opt}(k)$, i.e. $\boldsymbol{u}_{opt}(k)$, is given to the pulse-width modulator, which generates the switching signal $\boldsymbol{p}(k)$ and is then applied to the plant. At the next sampling instant $k+1$ the process is repeated. This method of feedback is known as the receding horizon policy.

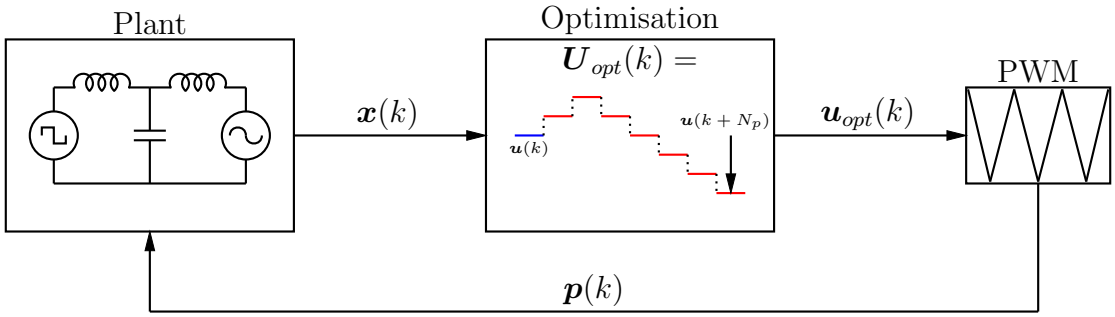


Figure 3.10: Block diagram depicting the receding horizon principle.

3.8 Quadratic programming formulation

The cost function in (3.40) forms a quadratic optimisation problem known as a *quadratic program* (QP). A typical QP can be expressed as

$$\begin{aligned} \text{minimise} \quad & \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{H}\boldsymbol{x} + \boldsymbol{q}^\top \boldsymbol{x} + r \\ \text{s.t.} \quad & \boldsymbol{G}\boldsymbol{x} \leq \boldsymbol{h} \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \ , \end{aligned} \tag{3.43}$$

where $\boldsymbol{G}\boldsymbol{x} \leq \boldsymbol{h}$ and $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ define the inequality- and equality constraints, respectively.[3] The goal of a QP is to minimise an quadratic function over a feasible set that forms a polyhedron [23]. With some manipulation (shown in Appendix C), (3.40) can be written in the same format as (3.43), yielding [9]

$$\begin{aligned} \boldsymbol{U}_{opt}(k) = \arg \underset{\boldsymbol{U}(k)}{\text{minimise}} \quad & \frac{1}{2}||\boldsymbol{U}(k)||^2_{\boldsymbol{H}} + \boldsymbol{\Theta}^\top(k)\boldsymbol{U}(k) + \theta(k) \ , \\ \text{s.t.} \quad & \boldsymbol{U}(k) \in \mathbb{U} \end{aligned} \tag{3.44}$$

where

[3] Note that the matrices and variables in (3.43) are in no way related to other matrices and variables previously defined. They are merely used to present the format of a typical QP.

$$\boldsymbol{H} = 2\{\boldsymbol{\Upsilon}\boldsymbol{Q}_c\boldsymbol{\Upsilon} + \lambda_u\boldsymbol{S}^\top\boldsymbol{S}\} \tag{3.45}$$

$$\boldsymbol{\Theta}^\top(k) = 2\Big\{[\boldsymbol{\Gamma}\boldsymbol{x}(k) - \boldsymbol{X}^*(k)]^\top\boldsymbol{Q}_c\boldsymbol{\Upsilon} + \boldsymbol{V}_g^\top(k)\boldsymbol{\Omega}^\top\boldsymbol{Q}_c\boldsymbol{\Upsilon} + \lambda_u[\boldsymbol{E}\boldsymbol{K}\boldsymbol{u}(k-1)]^\top\boldsymbol{S}\Big\} \tag{3.46}$$

$$\begin{aligned}\theta(k) = {} & ||\boldsymbol{\Gamma}\boldsymbol{x}(k) - \boldsymbol{X}^*(k)||^2_{\boldsymbol{Q}_c} + ||\boldsymbol{\Omega}\boldsymbol{V}_g(k)||^2_{\boldsymbol{Q}_c} + \lambda_u||\boldsymbol{E}\boldsymbol{u}(k)||_2^2 \\ & + 2[\boldsymbol{\Gamma}\boldsymbol{x} - \boldsymbol{X}^*(k)]^\top\boldsymbol{Q}_c\boldsymbol{\Omega}\boldsymbol{V}_g(k)\ .\end{aligned} \tag{3.47}$$

The matrix $\boldsymbol{H}$, known as the Hessian, is a symmetrical, positive definite matrix (for $\lambda_u > 0$) of second-order partial derivatives of scalar-valued cost function J. The Hessian does not include time-varying parameters, which means it can be calculated off-line and stored when implementing the optimisation in practise (if λ_u stays constant). The term $\theta(k)$ in (3.44) does not include the *optimiser* $\boldsymbol{U}(k)$, which means that it remains constant during optimisation and can, therefore, be omitted.

3.9 Optimisation

3.9.1 Gradient projection method

Since the Hessian is positive definite, there is only one global minimum of the objective function J. When relaxing the constraints in (3.44), the global minimum can be calculated by finding the point where the gradient of J is zero. If the gradient is written as

$$\nabla J(\boldsymbol{U}(k)) = \boldsymbol{H}\boldsymbol{U}(k) + \boldsymbol{\Theta}(k)\ , \tag{3.48}$$

then the unconstrained optimal sequence of PWM references $\boldsymbol{U}_{unc}(k)$ is calculated as

$$\boldsymbol{U}_{unc}(k) = -\boldsymbol{H}^{-1}\boldsymbol{\Theta}(k)\ . \tag{3.49}$$

Consider the example in Fig. 3.11. The dashed ellipses portray the level sets of J as contour lines. The first entry $\boldsymbol{u}_{unc}(k)$ of $\boldsymbol{U}_{unc}(k)$ violates the constraints and cannot be used directly as PWM reference input, as it does not fall within the feasible set $\mathbb{U}$. Using $\boldsymbol{u}_{unc}(k)$ directly would lead to over modulation of the pulse-width modulator. However, by making use of optimisation techniques, such as the gradient method (which will be discussed in Section 3.9), the optimal PWM reference $\boldsymbol{u}_{opt}(k)$ can be found.

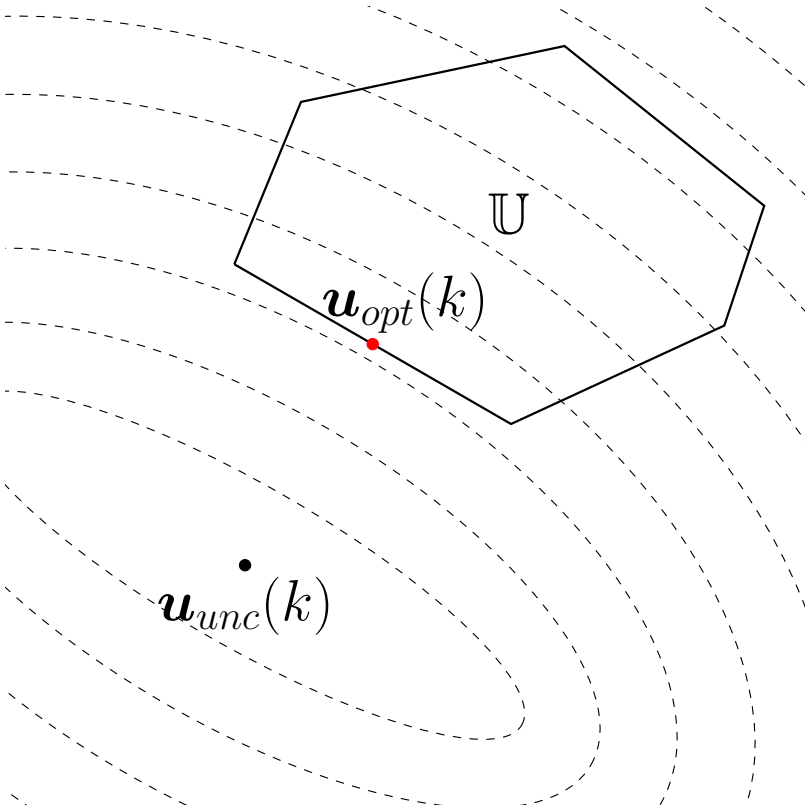


Figure 3.11: Depiction of QP solution space.

Three other optimisation methods were also considered to solve the optimisation problem, namely multi-parametric programming, active set methods, and interior point methods. According to [27], these methods either require large amounts of memory to store the controller (in case of multi-parametric programming), or high computational power when solved on-line. Multi-parametric programming and the active set methods were also reviewed in [28] and found to be either constrained to small-scale problems, or computationally taxing. A fourth solution method, the gradient projection method, is simple, yet robust and can be used for problems with a high number of dimensions without incurring a high computational burden. For this reason, the gradient projection method is chosen as optimisation technique in this thesis.

The gradient projection method uses the negative gradient $-\nabla J$ in order to point in the direction of the steepest descent. From any starting point, a step is taken in the steepest descent direction to provide the next iterate. When the next iterate falls within the infeasible region, it is projected back towards the feasible region. The step size requires the value of L, which is known as the Lipschitz continuous gradient of J, or the L-smoothness of J. Since the cost function J is a quadratic program, the value of L is set equal to the largest eigenvalue of $\boldsymbol{H}$ [25].

The solution space of the optimisation problem forms a hexagon in the $\alpha\beta$-plane since the optimiser $\boldsymbol{U}(k)$ consists of N_p $\alpha\beta$ entries. The optimisation stage, therefore, requires a complex projection method when compared to the simple box-constraints of the *abc* framework. The algorithm would have to perform a hyperplane search as well as a projection operator for every $\boldsymbol{u}$ in $\boldsymbol{U}$. See Appendix A for this approach. An alternative method is chosen, where each $\boldsymbol{u}$ is transformed back to *abc*, where a simple projection

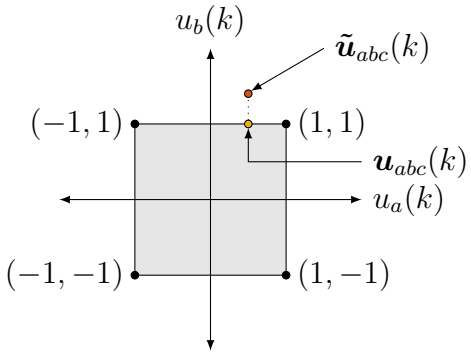


Figure 3.12: Projection step principle of the gradient method.

can be done using the box-constraints. The variables are transformed from $\alpha\beta$ to abc using $\boldsymbol{\xi}_{abc} = \boldsymbol{K}^{-1}\boldsymbol{\xi}_{\alpha\beta}$, where $\boldsymbol{K}^{-1} = 1.5\boldsymbol{K}^{\top}$ (see (3.4)). In Fig. 3.12, these constraints are visualised in a 2-D space, depicting only the a- and b-phases. In the figure, $\tilde{\boldsymbol{u}}_{abc}(k)$ denotes the non-projected reference vector and $\boldsymbol{u}_{abc}(k)$ denotes the orthogonal projection thereof onto the "box". Since the transformation assumes that the zero-sequence component is zero, a common-mode term is added to establish an appropriate zero-sequence.
Let $\boldsymbol{u}_{abc}(k) = [u_a(k)\ u_b(k)\ u_c(k)]^{\top}$ denote the modulating vector at the current iteration of the optimisation step. As discussed in [29], the SVM-type common-mode component can be added to $\boldsymbol{u}(k)$, with

$$\boldsymbol{u}_{svm}(k) = \boldsymbol{u}_{abc}(k) - \frac{\max(\boldsymbol{u}_{abc}(k)) + \min(\boldsymbol{u}_{abc}(k))}{2}[1\ 1\ 1]^{\top}. \tag{3.50}$$

The addition of the common-mode term is indicated as the SVM-operator in Algorithm 1.

Algorithm 1 Gradient Projection Algorithm

function GRADIENT PROJECTION METHOD($\boldsymbol{H}, \boldsymbol{\Theta}, \boldsymbol{u}_0, L$)
  for $i = 1 : N_f$ **do** ▷ For a predetermined number of iterations.
    $\boldsymbol{U}_i \leftarrow \boldsymbol{U}_{i-1} - \frac{1}{L}(\boldsymbol{H}\boldsymbol{U}_{i-1} + \boldsymbol{\Theta})$ ▷ Step in steepest descent.
    for $n = 1 : N_p$ **do** ▷ For each $\boldsymbol{u}$ in $\boldsymbol{U}$.
      $\tilde{\boldsymbol{u}}_{abc}(n) \leftarrow \text{SVM}(\boldsymbol{K}^{-1}\boldsymbol{u}(n))$ ▷ Transform to abc and add common-mode term.
      for $k = 1 : 3$ **do** ▷ Project $\tilde{\boldsymbol{u}}_{abc}$ onto box.
        $\boldsymbol{u}_{abc}(k) \leftarrow \tilde{\boldsymbol{u}}_{abc}(k)$
        if $\boldsymbol{u}_{abc}(k) \geq 1$ **then**
          $\boldsymbol{u}_{abc}(k) \leftarrow 1$
        else if $\boldsymbol{u}_{abc}(k) \leq -1$ **then**
          $\boldsymbol{u}_{abc}(k) \leftarrow -1$
        end if
      end for
      $\boldsymbol{u}(n) \leftarrow \boldsymbol{K}\boldsymbol{u}_{abc}(n)$ ▷ Transform back to $\alpha\beta$ reference frame.
    end for
  end for
end function

3.9.2 Implementation feasibility

The Digilent ZedBoard XC7Z020 development board, which contains the Xilinx Zynq-7020 SoC architecture, is considered for implementation of the controller. The relatively low-cost Zynq-7020 includes both a dual-core ARM Cortex-A9 processing system and a 28 nm Xilinx FPGA. Only the latter will be used for the controller. The FPGA contains 220 digital signal processing (DSP) slices, which each include an 18×25 signed multiplier and a 48-bit adder/accumulator [30].

The controller executes at twice the switching frequency $f_s = 1650\,\text{Hz}$ of the converter, which means that the total allowed calculation time is 303 µs. The calculation of $\boldsymbol{\Theta}(k)$ and the gradient projection algorithm are the largest contributors to resource/time usage, since both require a high number of multiplications. There exists a trade-off between the amount of required FPGA resources and the amount of time it takes those resources to execute operations, e.g. for 100 multiplications, 10 multipliers might require 10 clock cycles, whereas 100 multipliers would require a single clock cycle.

After rearranging the matrices within (3.46) and taking into account that certain matrices can be calculated off-line, the number of on-line multiplications needed for $\boldsymbol{\Theta}(k)$ is $16N_p + 18N_p^2$, where N_p denotes the prediction horizon. The gradient projection method requires $N_f(4N_p^2 + 12N_p)$ multiplications, where N_f denotes the amount of iterations of the gradient method. This means a total number of $N_p^2(4N_f + 18) + N_p(12N_f + 16)$ multiplications per sampling interval. The optimisation algorithm is limited to $N_f = 50$ iterations. It is assumed that each multiplication is executed at every consecutive clock cycle (at 10 MHz only for this example). Table 3.6 shows estimates of time required for the controller to execute for different values of N_p.

Table 3.6: Control algorithm execution times

	Multipliers				
N_p	1	10	50	100	220
1	83.4 µs	8.34 µs	1.67 µs	0.83 µs	379 ns
4	595 µs	59.5 µs	11.9 µs	5.95 µs	2.7 µs
8	1.9 ms	188.8 µs	37.8 µs	18.9 µs	8.58 µs
12	3.9 ms	387.8 µs	77.6 µs	38.8 µs	17.6 µs
16	6.6 ms	656.6 µs	131.3 µs	65.7 µs	29.8 µs

3.10 Alternative method

3.10.1 Introduction to LQR controller

An alternative approach that will be investigated in this thesis, is the trajectory-based linear quadratic regulator (LQR) controller, developed in [8]. Since an LQR controller can be applied to systems with linear dynamics and requires a quadratic cost function, the proposed method seems fitting to the control problem at hand. The simplicity of the LQR design also makes it an attractive approach. One disadvantage of the LQR

controller, is that it does not take into account constraints.

The design of the controller in this section is entirely based on [8]. The topology of the system is depicted in Fig. 4.1, Section 4.2, where the the internal dynamic model of the system is also derived. The topology consists out of a two-level VSI connected to the grid via an *LCL*-filter. The proposed controller is illustrated as a block diagram in Fig. 3.13.

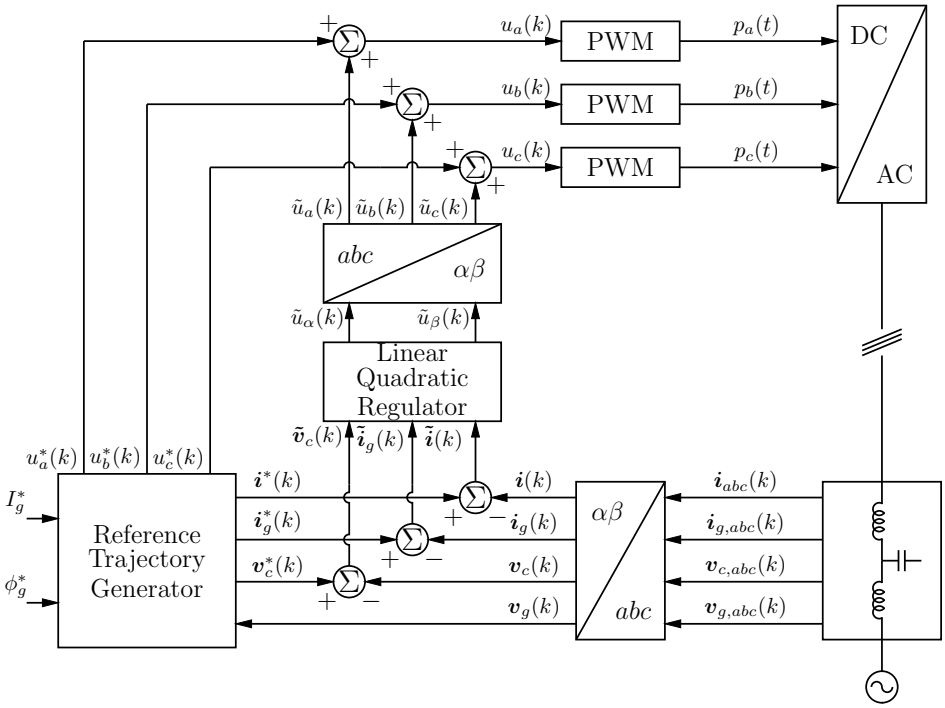


Figure 3.13: Block diagram depicting trajectory-based LQR controller (reproduced from [8]).

The state vector is chosen as

$$\boldsymbol{x} = [\boldsymbol{i}^\top \ \boldsymbol{i}_g^\top \ \boldsymbol{v}_c^\top]^\top \ , \tag{3.51}$$

where $\boldsymbol{i}$ denotes the inverter currents, $\boldsymbol{i}_g$ denotes the grid currents, and $\boldsymbol{v}_c$ denotes the filter capacitor voltages. The state vector is defined in $\alpha\beta$ format. The control principle of the controller can be summarized as follows:

- "Reference Trajectory" Generator generates steady-state trajectory of state vector $\boldsymbol{x}^*(k)$ as well as three-phase PWM reference signals $\boldsymbol{u}^*_{abc}(k)$ according to the grid current amplitude I_g^* and phase ϕ_g^*.
- Difference between measured and reference signals is fed to the LQR, which regulates these small-signal differences to zero.
- The LQR output is added to the PWM reference signals that adjust the output pulses of $\boldsymbol{p}_{abc}(t)$.

The design of the controller requires two steps, namely the calculation of the steady-state reference signals and secondly the derivation of the small-signal model used by the LQR.

3.10.2 Calculation of steady-state reference signals

The reference signal is sampled at asymmetrical regular intervals, as can be seen in Fig. 3.14.

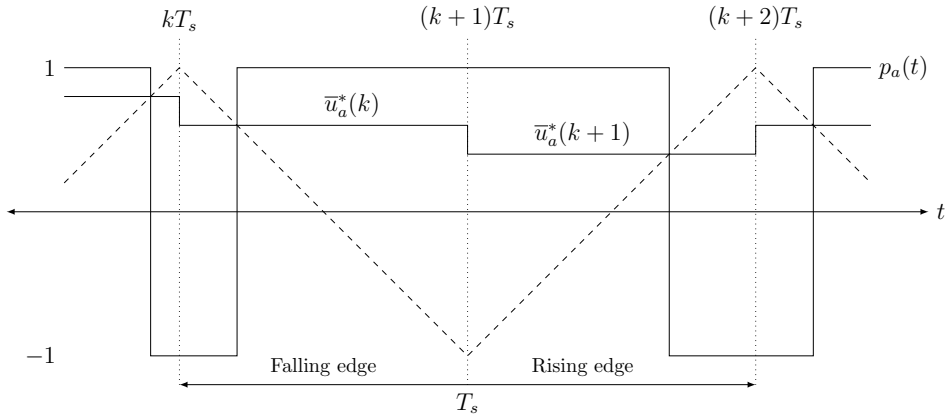


Figure 3.14: Illustration of PWM waveforms for LQR controller. The dotted lines indicate the sampling instants $(k+i)T_s$. Replicated from [8].

It is assumed that the system in Fig. 4.1 is in steady state for the calculation of the reference signals. The first step is to calculate the PWM reference signals. The fundamental of the a-phase grid current $i_{g,a}(t)$ can be calculated using the specified values of I_g and ϕ_g. The fundamental component of the inverter output voltage $v_a(t)$ can then be calculated with phasor analysis. The a-phase PWM reference signal can now be given as

$$u_a^*(kT_s) = 2\frac{V_a}{V_d}\sin(\omega_1 kT_s + \phi_a) \ , \tag{3.52}$$

where ω_1 denotes the fundamental angular grid frequency, V_d denotes the DC bus voltage, and V_a and ϕ_a denotes the amplitude and phase of the fundamental component of the output voltage $v_a(t)$, respectively. Since asymmetric regular sampling is used, there is a slight difference between the amplitude and phase of $u_a^*(kT_s)$ and that of the fundamental component of $p_a(t)$. This is addressed by phase shifting the references by $\frac{T_s}{2}$ and adjusting the amplitude according to equation (6.63) in [17]. The b- and c phase references have the same magnitude as a but lag $u_a^*(kT_s)$ with 120° and 240°, respectively. Space-vector centering is then applied to $\boldsymbol{u}_{abc}^*(kT_s)$ by adding the common mode term in (3.18), yielding the new PWM reference signals $\overline{\boldsymbol{u}}_{abc}^*(k)$.

The second step is the calculation of the steady-state trajectory of the state vector in (3.51). The principle of superposition is used to calculate the effects of the inverter and grid voltages separately. Firstly, the grid voltages $\boldsymbol{v}_{g,abc}(k)$ are assumed to be zero. The carrier frequency f_c is chosen as a multiple of three times the fundamental frequency f_1 of the grid, causing periodicity of the state variables in the steady-state. The steady-state trajectory of $\boldsymbol{x}(k)$ is calculated by first setting $k=1$ and integrating (3.21) over the first half of the switching period where the carrier is falling. This leads to

$$\boldsymbol{x}(k+1) = \boldsymbol{e}^{T_s\boldsymbol{F}}\boldsymbol{x}(k) + \boldsymbol{\mathcal{H}}_f(\overline{u}_a(k), \overline{u}_b(k), \overline{u}_c(k)) \ , \tag{3.53}$$

where

$$\boldsymbol{\mathcal{H}}_f(\overline{u}_a(k),\overline{u}_b(k),\overline{u}_c(k)) = 2\boldsymbol{F}^{-1}\boldsymbol{e}^{\frac{T_s}{2}\boldsymbol{F}}\Big\{\boldsymbol{e}^{\frac{T_s}{2}\overline{u}_a(k)\boldsymbol{F}}\boldsymbol{G}[1\ 0\ 0]^\top + \boldsymbol{e}^{\frac{T_s}{2}\overline{u}_b(k)\boldsymbol{F}}\boldsymbol{G}[0\ 1\ 0]^\top + \boldsymbol{e}^{\frac{T_s}{2}\overline{u}_c(k)\boldsymbol{F}}\boldsymbol{G}[0\ 0\ 1]^\top\Big\} - \boldsymbol{F}^{-1}[\boldsymbol{e}^{T_s\boldsymbol{F}} + \boldsymbol{I}]\boldsymbol{G}[1\ 1\ 1]^\top\ . \tag{3.54}$$

The integration of (3.21) over the second half carrier period (rising edge), results in

$$\boldsymbol{x}(k+1) = \boldsymbol{e}^{T_s\boldsymbol{F}}\boldsymbol{x}(k) + \boldsymbol{\mathcal{H}}_r(\overline{u}_a(k),\overline{u}_b(k),\overline{u}_c(k))\ , \tag{3.55}$$

where

$$\boldsymbol{\mathcal{H}}_r(\overline{u}_a(k),\overline{u}_b(k),\overline{u}_c(k)) = -2\boldsymbol{F}^{-1}\boldsymbol{e}^{\frac{T_s}{2}\boldsymbol{F}}\Big\{\boldsymbol{e}^{-\frac{T_s}{2}\overline{u}_a(k)\boldsymbol{F}}\boldsymbol{G}[1\ 0\ 0]^\top + \boldsymbol{e}^{-\frac{T_s}{2}\overline{u}_b(k)\boldsymbol{F}}\boldsymbol{G}[0\ 1\ 0]^\top + \boldsymbol{e}^{-\frac{T_s}{2}\overline{u}_c(k)\boldsymbol{F}}\boldsymbol{G}[0\ 0\ 1]^\top\Big\} + \boldsymbol{F}^{-1}[\boldsymbol{e}^{T_s\boldsymbol{F}} + \boldsymbol{I}]\boldsymbol{G}[1\ 1\ 1]^\top\ . \tag{3.56}$$

In order to calculate the values of $\boldsymbol{x}(k)$ over a full period, the first sample $\boldsymbol{x}(1)$ is needed. In order to calculate this value, the equations (3.53) and (3.55) are applied over one fundamental period, yielding

$$\begin{aligned}\boldsymbol{x}(N) &= \boldsymbol{e}^{NT_s\boldsymbol{F}}\boldsymbol{x}(1) + \boldsymbol{e}^{(N-1)T_s\boldsymbol{F}}\boldsymbol{\mathcal{H}}_f(\overline{u}_a(1),\overline{u}_b(1),\overline{u}_c(1)) + \boldsymbol{e}^{(N-2)T_s\boldsymbol{F}}\boldsymbol{\mathcal{H}}_r(\overline{u}_a(2),\overline{u}_b(2),\overline{u}_c(2))\\ &\quad + \cdots + \boldsymbol{e}^{T_s\boldsymbol{F}}\boldsymbol{\mathcal{H}}_f(\overline{u}_a(N-1),\overline{u}_b(N-1),\overline{u}_c(N-1)) + \boldsymbol{\mathcal{H}}_r(\overline{u}_a(N),\overline{u}_b(N),\overline{u}_c(N))\\ &= \boldsymbol{e}^{NT_s\boldsymbol{F}}\boldsymbol{x}(1) + \boldsymbol{\mathcal{S}}\ .\end{aligned} \tag{3.57}$$

Since the reference signals are periodic, $\boldsymbol{x}(N) = \boldsymbol{x}(1)$ holds true and, therefore

$$\boldsymbol{x}(1) = [\boldsymbol{I} - \boldsymbol{e}^{NT_s\boldsymbol{F}}]^{-1}\boldsymbol{\mathcal{S}}\ . \tag{3.58}$$

Now that $\boldsymbol{x}(1)$ is known, the steady-state trajectory of the state variables can be calculated over a fundamental period, by using (3.53) and (3.55).

Finally, the effect of the grid voltages is calculated by assuming the output voltages of the inverter to be zero and using phasor analysis. The state reference vector $\boldsymbol{x}^*(k)$ is found by adding the effects of the grid voltage to the steady-state trajectory calculated above.

3.10.3 Derivation of small-signal model in discrete time

Consider the adjusted PWM signal waveforms in Fig. 3.15. The purpose of the LQR controller is to make small adjustments to the PWM reference signals $\overline{\boldsymbol{u}}^*(k)$ in order to drive $\boldsymbol{x}$ into steady-state. The output of the PWM waveform changes with the addition of the small-signal $\tilde{u}_a(t)$ values. The difference between the original and modified pulse sequence, $p_a(t)$ and $\tilde{p}_a(t)$, forms narrow rectangles, which can be modelled by impulses

as shown in the figure. The magnitude of each impulse is equal to the area of the corresponding narrow rectangle. It is assumed that each impulse occurs at a half of the sampling period, i.e. every $(k+\frac{1}{2})T_s$ seconds.

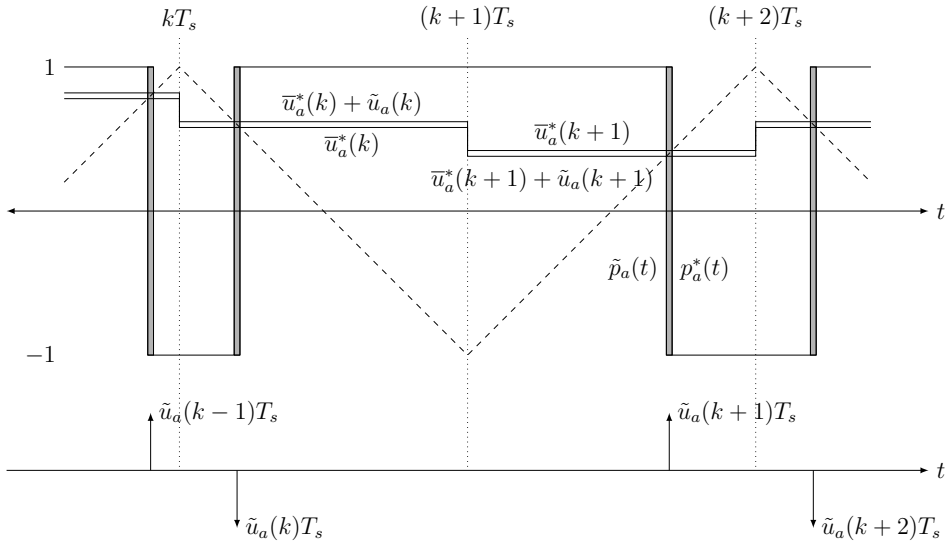


Figure 3.15: Illustration of small-signal PWM waveforms for LQR controller. Narrow rectangular pulses are approximated by impulses. Replicated from [8].

To derive the small signal model, it is assumed that the output of the "Reference Trajectory Generator" in Fig. 3.13 as well as the grid voltages are zero. The pulse-width-modulators of each phase arm are replaced with the impulses. The small-signal state vector is defined as

$$\tilde{\boldsymbol{x}}(k) = [\tilde{\boldsymbol{i}}^{\top}(k)\ \tilde{\boldsymbol{i}}_g^{\top}(k)\ \tilde{\boldsymbol{i}}_c^{\top}(k)]^{\top}\ . \tag{3.59}$$

If $\tilde{\boldsymbol{u}}(k)$ denotes the small-signal reference vector in $\alpha\beta$ and $\delta(t-t_0)$ represents an impulse of strength 1 at time t_0, then small-signal state-vector update equation can be written as

$$\begin{aligned}\tilde{\boldsymbol{x}}(k+1) &= \boldsymbol{e}^{T_s\boldsymbol{F}}\tilde{\boldsymbol{x}}(k) + T_s\int_0^{T_s}\boldsymbol{e}^{(T_s-\tau)\boldsymbol{F}}\delta(\tau-\frac{T_s}{2})\boldsymbol{G}\tilde{\boldsymbol{u}}(k)d\tau \\ &= \boldsymbol{e}^{T_s\boldsymbol{F}}\tilde{\boldsymbol{x}}(k) + T_s\boldsymbol{e}^{\frac{T_s}{2}\boldsymbol{F}}\boldsymbol{G}\tilde{\boldsymbol{u}}(k) \\ &= \boldsymbol{A}_{\int}\tilde{\boldsymbol{x}}(k) + \boldsymbol{B}_{\int}\tilde{\boldsymbol{u}}(k)\ ,\end{aligned} \tag{3.60}$$

where

$$\boldsymbol{A}_{\int} = \boldsymbol{e}^{T_s\boldsymbol{F}} \text{ and } \boldsymbol{B}_{\int} = T_s\boldsymbol{e}^{\frac{T_s}{2}\boldsymbol{F}}\boldsymbol{G}\ .$$

3.10.4 LQR cost function

The small-signal LQR controller cost function is defined as

$$J = \sum_{k=1}^{\infty}\tilde{\boldsymbol{x}}^{\top}(k)\boldsymbol{Q}\tilde{\boldsymbol{x}}(k) + \tilde{\boldsymbol{u}}^{\top}(k)\boldsymbol{R}\tilde{\boldsymbol{u}}(k) \tag{3.61}$$

where $\boldsymbol{Q}$ is a weight matrix applied to the different state variables and $\boldsymbol{R}$ is a penalty on the change in $\tilde{\boldsymbol{u}}$. The feedback gain K can easily be calculated with the MATLAB function `dlqr()`. For implementation, the PWM reference signals $\overline{\boldsymbol{u}}^*_{abc}(k)$ and the state reference vector $\boldsymbol{x}^*(k)$ are calculated off-line and stored in lookup tables on an FPGA.

3.11 Summary

This chapter gave the reader an overview on fundamental concepts, hardware topologies and controller principles necessary for the remainder of this thesis.

The topology of the relevant inverter was discussed in the first part of this chapter, where the mathematical model was also defined. The relevant background information was given, relating to the *LCL*-filter, the power grid and modulation techniques. Two main control strategies were explored, namely the proposed controller and the small-signal LQR controller. The implementation of these control strategies are further explored in the following chapters.

Chapter 4

Controller Design

4.1 Introduction

This chapter discusses the process of developing a control algorithm to suit the control requirements of the given topology. The system topology is firstly discussed and an internal dynamic model is derived. The cost function is formulated and the control algorithm is presented.

4.2 System Model

Consider Fig. 4.1. The plant consists of a two-level VSI, connected to the grid (represented by three-phase voltage sources), via an *LCL*-filter. The figure shows inverter currents (i_a, i_b and i_c), the grid-side currents (i_{ga}, i_{gb} and i_{gc}), the filter capacitor voltages (v_{ca}, v_{cb} and v_{cc}) and the neutral point n between the two bus capacitors (represented by two DC voltage sources). The neutral point is mainly conceptual in order to help with per-phase model derivations. The equivalent series resistances (ESR) of the filter inductors (R), the filter capacitors (R_c) and the grid inductances (R_g) are also included in the system model.

The inverter currents $\boldsymbol{i}_{abc}$, grid currents $\boldsymbol{i}_{g,abc}$ and filter capacitor voltages $\boldsymbol{v}_{c,abc}$ are chosen as the state variables. The system model is deduced by transforming these variables from abc to the $\alpha\beta$ reference frame by using the Clarke transformation $\boldsymbol{\zeta}_{\alpha\beta} = \boldsymbol{K}\boldsymbol{\zeta}_{abc}$. The state vector is defined as

$$\boldsymbol{x} = \begin{bmatrix} i_\alpha & i_\beta & i_{g\alpha} & i_{g\beta} & v_{c\alpha} & v_{c\beta} \end{bmatrix}^\top . \tag{4.1}$$

To simplify the notation, the $\alpha\beta$ subscripts are dropped and will only be included when the notations are ambiguous. The output voltages of a single phase arm is given by $v_x = \frac{V_d}{2} p_x$, with $x \in \{a, b, c\}$ and $p_x \in \{-1, 1\}$. The continuous-time state-space equation that describe the system is

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{F}\boldsymbol{x}(t) + \boldsymbol{G}\boldsymbol{p}(t) + \boldsymbol{P}\boldsymbol{v}_{g,abc}(t) \ , \tag{4.2}$$

with $\boldsymbol{p}(t) = [p_\alpha(t) \ p_\beta(t)]^\top$ and $\boldsymbol{v}_{g,abc}(t) = [v_{ga}(t) \ v_{gb}(t) \ v_{gc}(t)]^\top$ denoting the switch positions in $\alpha\beta$ and the grid voltages, respectively. The matrices $\boldsymbol{F}$, $\boldsymbol{G}$ and $\boldsymbol{P}$ are given by

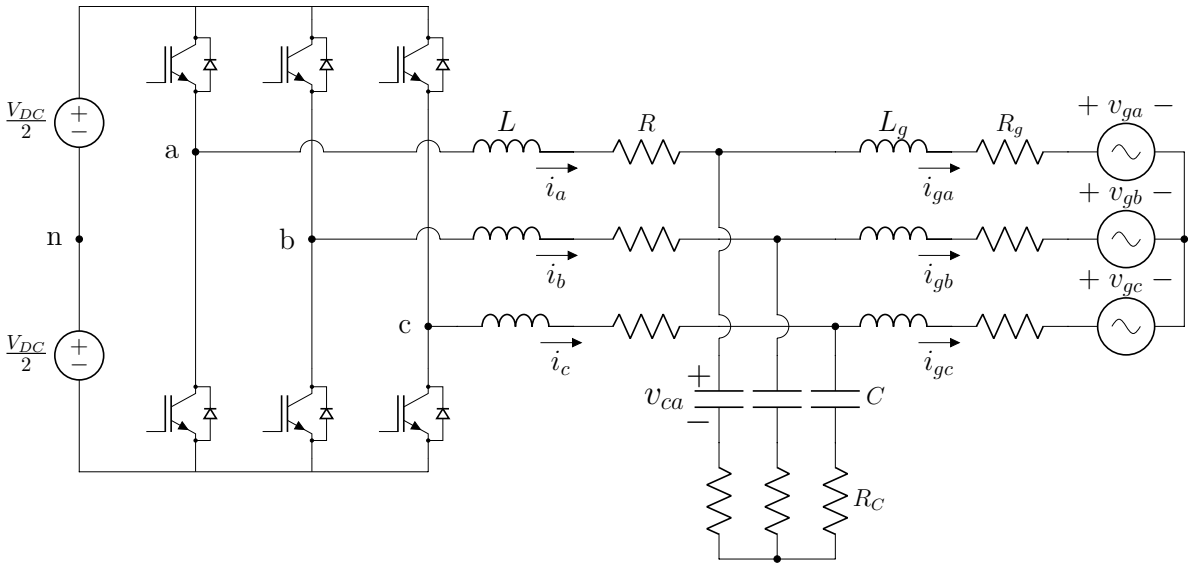


Figure 4.1: Topology of grid-connected inverter via an LCL-filter

$$\boldsymbol{F} = \begin{bmatrix} \frac{R+R_C}{-L} & 0 & \frac{R_C}{L} & 0 & \frac{1}{-L} & 0 \\ 0 & \frac{R+R_C}{-L} & 0 & \frac{R_C}{L} & 0 & \frac{1}{-L} \\ \frac{R_C}{L_g} & 0 & \frac{R_g+R_C}{-L_g} & 0 & \frac{1}{L_g} & 0 \\ 0 & \frac{R_C}{L_g} & 0 & \frac{R_g+R_C}{-L_g} & 0 & \frac{1}{L_g} \\ \frac{1}{C} & 0 & \frac{1}{-C} & 0 & 0 & 0 \\ 0 & \frac{1}{C} & 0 & \frac{1}{-C} & 0 & 0 \end{bmatrix} , \tag{4.3}$$

$$\boldsymbol{G} = \begin{bmatrix} \frac{V_d}{2L} & 0 \\ 0 & \frac{V_d}{2L} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \text{ and } \boldsymbol{P} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{-L_g} & 0 \\ 0 & \frac{1}{-L_g} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \boldsymbol{K} . \tag{4.4}$$

The matrix $\boldsymbol{P}$ and vector $\boldsymbol{v}_{g,abc}(t)$ represent the effect of the grid on the system. The discrete-time model of the system is derived using exact discretization of the continuous-time model (see Section 3.6.2), resulting in

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{p}(k) + \boldsymbol{\mathcal{V}}\boldsymbol{v}_{g,abc}(k) , \tag{4.5}$$

where

$$\boldsymbol{A} = e^{\boldsymbol{F}T_s} \tag{4.6}$$

$$\boldsymbol{B} = -\boldsymbol{F}^{-1}(\boldsymbol{I}_{n_x} - \boldsymbol{A})\boldsymbol{G} \tag{4.7}$$

$$\boldsymbol{\mathcal{V}} = -\boldsymbol{F}^{-1}(\boldsymbol{I}_{n_x} - \boldsymbol{A})\boldsymbol{P} . \tag{4.8}$$

The derivation of (4.2) is presented in Appendix B.

4.3 Control Problem

The goal of the controller is to generate PWM references for a pulse-width modulator. The PWM reference signal vector $\boldsymbol{u}(k) = [u_\alpha \; u_\beta]^\top$ is introduced, where $\boldsymbol{u}(k) \in [-1, 1]^2$. The tracking error between the reference signals $\boldsymbol{x}^*(k)$ and state variables $\boldsymbol{x}(k)$ as well as a weight λ_u on the change in PWM reference signal $\boldsymbol{\Delta u}(k)$ are included as control objectives in the cost function. The value of λ_u sets the trade-off between tracking accuracy and control effort. The cost function can be written as

$$J(\boldsymbol{x}(k), \boldsymbol{x}^*(k), \boldsymbol{U}(k)) = \sum_{\ell=k}^{k+N_p-1} ||\boldsymbol{x}^*(\ell+1) - \boldsymbol{x}(\ell+1)||^2_{\boldsymbol{Q}} + \lambda_u ||\Delta \boldsymbol{u}(\ell)||_2^2 \; , \tag{4.9}$$

The change in PWM reference signal $\boldsymbol{\Delta u}(k)$ is defined as the difference between two consecutive PWM reference vectors $\boldsymbol{\Delta u}(\ell) = \boldsymbol{u}(\ell) - \boldsymbol{u}(\ell-1)$. A single PWM reference vector is represented by $\boldsymbol{u}(k)$. The sequence of PWM reference vectors with size $2N_p \times 1$ is defined over the prediction horizon N_p as

$$\boldsymbol{U}(k) = [\boldsymbol{u}^\top(k) \; \boldsymbol{u}^\top(k+1) \cdots \boldsymbol{u}^\top(k+N_p-1)]^\top \; . \tag{4.10}$$

The optimal sequence of PWM reference vectors can be found through minimization of the cost function. The optimisation problem is stated as

$$\begin{aligned} \boldsymbol{U}_{opt}(k) = \arg \underset{\boldsymbol{U}(k)}{\text{minimise}} \; & J(\boldsymbol{x}(k), \boldsymbol{x}^*(k), \boldsymbol{U}(k)) \\ \text{s.t.} \; & \boldsymbol{U}(k) \in \mathbb{U} \; , \end{aligned} \tag{4.11}$$

where $\mathbb{U} = \boldsymbol{\mathcal{U}}^{N_p}$. As explained in Section 3.8, the optimisation problem can be formulated as a quadratic program of the form

$$\begin{aligned} \boldsymbol{U}_{opt}(k) = \arg \underset{\boldsymbol{U}(k)}{\text{minimise}} \; & \frac{1}{2}||\boldsymbol{U}(k)||^2_{\boldsymbol{H}} + \boldsymbol{\Theta}^\top(k)\boldsymbol{U}(k) \; , \\ \text{s.t.} \; & \boldsymbol{U}(k) \in \mathbb{U} \; , \end{aligned} \tag{4.12}$$

where

$$\boldsymbol{H} = 2\{\boldsymbol{\Upsilon}\boldsymbol{Q}_c\boldsymbol{\Upsilon} + \lambda_u \boldsymbol{S}^\top \boldsymbol{S}\} \text{ and} \tag{4.13}$$

$$\boldsymbol{\Theta}^\top(k) = 2\Big\{ [\boldsymbol{\Gamma}\boldsymbol{x}(k) - \boldsymbol{X}^*(k)]^\top \boldsymbol{Q}_c \boldsymbol{\Upsilon} + \boldsymbol{V}_g^\top(k) \boldsymbol{\Omega}^\top \boldsymbol{Q}_c \boldsymbol{\Upsilon} + \lambda_u [\boldsymbol{E}\boldsymbol{K}\boldsymbol{u}(k-1)]^\top \boldsymbol{S} \Big\} \; . \tag{4.14}$$

The matrices that make up $\boldsymbol{\Theta}^\top(k)$ are defined as

$$\boldsymbol{\Gamma} = \begin{bmatrix} (\boldsymbol{A})^\top & (\boldsymbol{A}^2)^\top & \cdots & (\boldsymbol{A}^{N_p})^\top \end{bmatrix}^\top \; , \tag{4.15}$$

$$\boldsymbol{\Upsilon} = \begin{bmatrix} \boldsymbol{B} & 0 & \cdots & 0 \\ \boldsymbol{AB} & \boldsymbol{B} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ \boldsymbol{A}^{N_p-1}\boldsymbol{B} & \boldsymbol{A}^{N_p-2}\boldsymbol{B} & \cdots & \boldsymbol{B} \end{bmatrix} \text{ and } \boldsymbol{\Omega} = \begin{bmatrix} \boldsymbol{\mathcal{V}} & 0 & \cdots & 0 \\ \boldsymbol{A\mathcal{V}} & \boldsymbol{\mathcal{V}} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ \boldsymbol{A}^{N_p-1}\boldsymbol{\mathcal{V}} & \boldsymbol{A}^{N_p-2}\boldsymbol{\mathcal{V}} & \cdots & \boldsymbol{\mathcal{V}} \end{bmatrix} \; . \tag{4.16}$$

The proposed controller setup is presented in Fig. 4.2. Measurements are taken from the plant and converted from abc to $\alpha\beta$ reference frame. The phase of the grid voltages is obtained by means of a phase-locked loop (PLL) for the purpose of synchronisation. The grid current amplitude I_g^* and phase ϕ_g^* are chosen and the reference signal vector $\boldsymbol{X}^*(k)$ is calculated. For a practical system, the values of I_g^* and ϕ_g^* will typically be generated by a PI control loop. The model predictive controller receives the state variables $\boldsymbol{x}(k)$ as well as the state reference vector $\boldsymbol{X}^*(k)$ as inputs and determines the optimal sequence of PWM reference signals $\boldsymbol{U}_{opt}(k)$ over a prediction horizon. A common-mode term is added to the first entry $\boldsymbol{u}(k)$ in the sequence, which is then applied to a pulse-width modulator to produce the switching signals $\boldsymbol{p}(k)$.

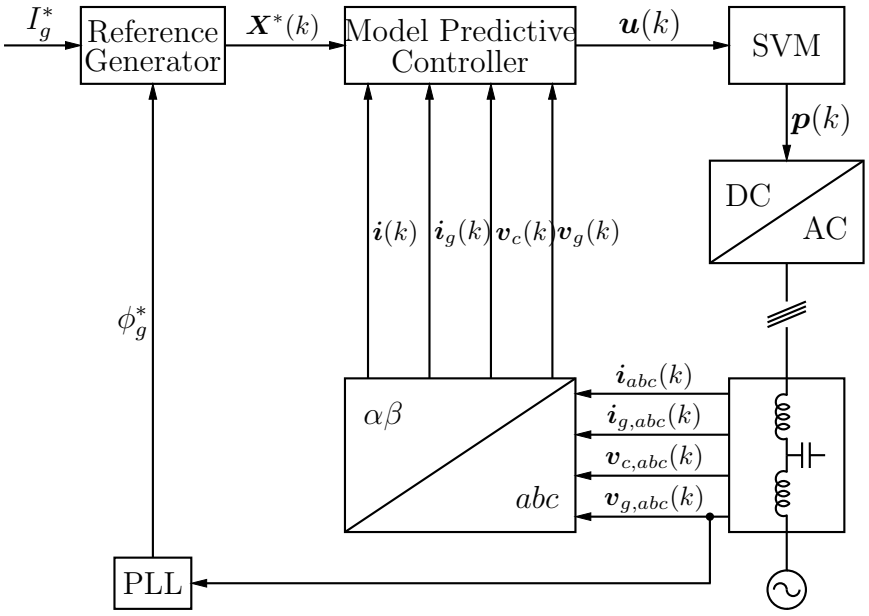


Figure 4.2: Flow diagram of proposed controller.

4.4 Summary

This chapter focuses on the controller design for the gradient projection method. The internal dynamic model of the inverter and LCL-filter pair is defined, after which the cost function is derived. A flow diagram depicting the proposed controller is finally discussed. The controller for the small-signal LQR method is derived in [8].

Chapter 5

Simulation Results

5.1 Introduction

In order to test the feasibility of each method, the controllers are tested in MATLAB-based simulations. The results of the proposed controller and the small-signal LQR method are presented and discussed. The Per Unit (PU) system is used to normalise the values of the state variables. The use of the PU system is advantageous when it comes to implementation on an FPGA, since it simplifies the use of fixed point calculations.

The first part of this chapter focuses on the simulation results of the proposed controller. The controller's response to a step in grid current reference is first examined and discussed, after which the robustness of the controller is tested by reducing the grid inductance. The grid current spectrum is analysed and different weighting factors and prediction horizons are tested to find the optimal combination.

In the second part of this chapter, the simulation results of the small-signal LQR method is examined. A step in grid current reference is once again applied to see the transient response of the controller. Thorough simulation results of this controller can be found in [8].

5.2 Gradient Projection Method

The mathematical system model of the plant is realised in a Matlab-based simulation. The system parameters are given in Table 5.1.

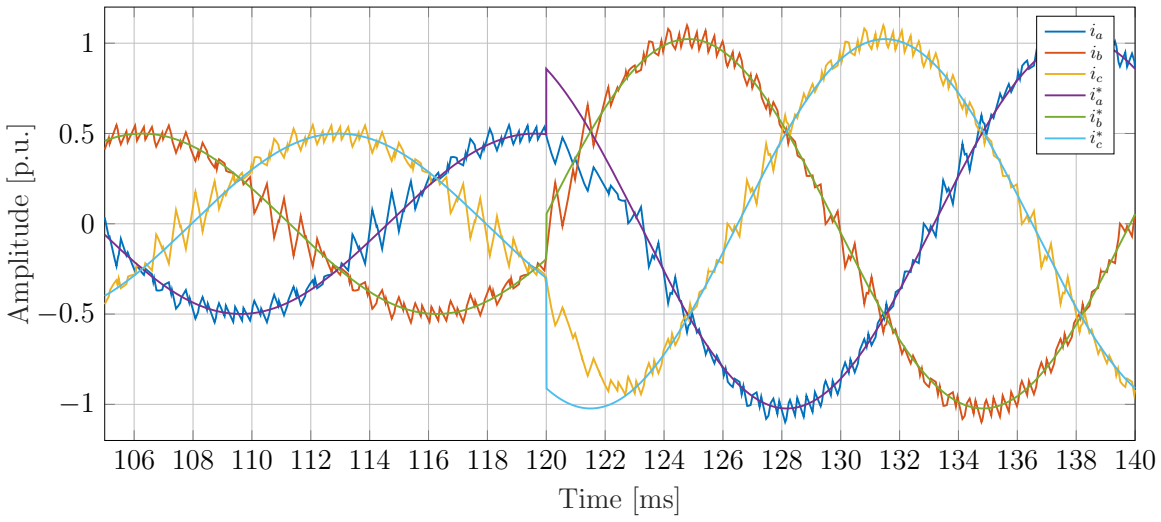
The nominal grid current is chosen as $I_g = 4132\,\text{A}$ (rms) and a unity power factor is chosen, i.e. at $\phi_g = 0°$, at the point of common coupling (PCC). The other reference signals are generated by using phasor analysis. The penalty matrix is chosen as $\boldsymbol{Q} = \text{diag}\ (0.2\ \ 0.2\ \ 1\ \ 1\ \ 0.1\ \ 0.1)$ in order to prioritise the grid-side current. The resonant frequency of the transfer function from the converter output voltage to the grid current is calculated as $f_{res,1} = 1831.98\,\text{Hz}$. The resonant frequency of the transfer function from the converter output voltage to the converter current is at $f_{res,2} = 536.9\,\text{Hz}$.

Table 5.1: System parameters for simulation.

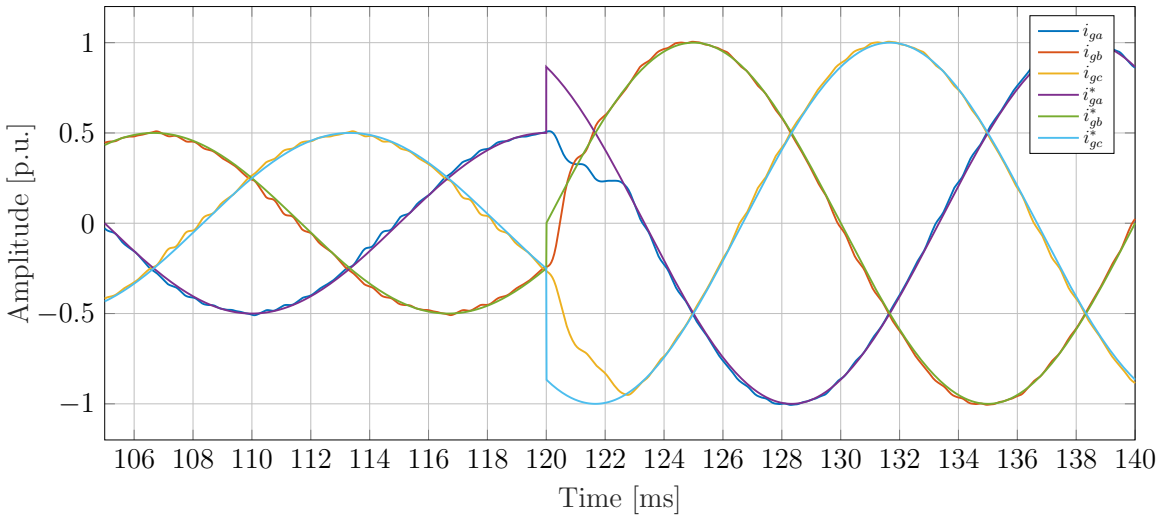
Parameter	Symbol	Value
Filter inductance	L	68 µH
Grid inductance	L_g	44.38 µH
Filter capacitance	C	1.98 mF
Filter inductor ESR	R	0.54 mΩ
Grid inductance ESR	R_g	1.76 mΩ
Filter capacitor ESR	R_C	0.67 mΩ
Prediction horizon	N_p	14
Input change penalty	λ_u	6×10^4
DC-link voltage	V_d	1.05 kV
Grid voltage (line-to-line)	V_g	690 V (rms)
Fundamental grid frequency	f_1	50 Hz
Switching frequency	f_c	1.65 kHz
Simulation sampling interval	T_s	606.07 ns

The base values for the PU system are $V_{base} = \sqrt{2}V_g$, $I_{base} = \sqrt{2}I_g$ and $\omega_b = f_1$. In order to show the response of the controller during transients, a step in the grid-current reference signals is introduced at $t = 120$ ms from 0.5 to 1 PU.

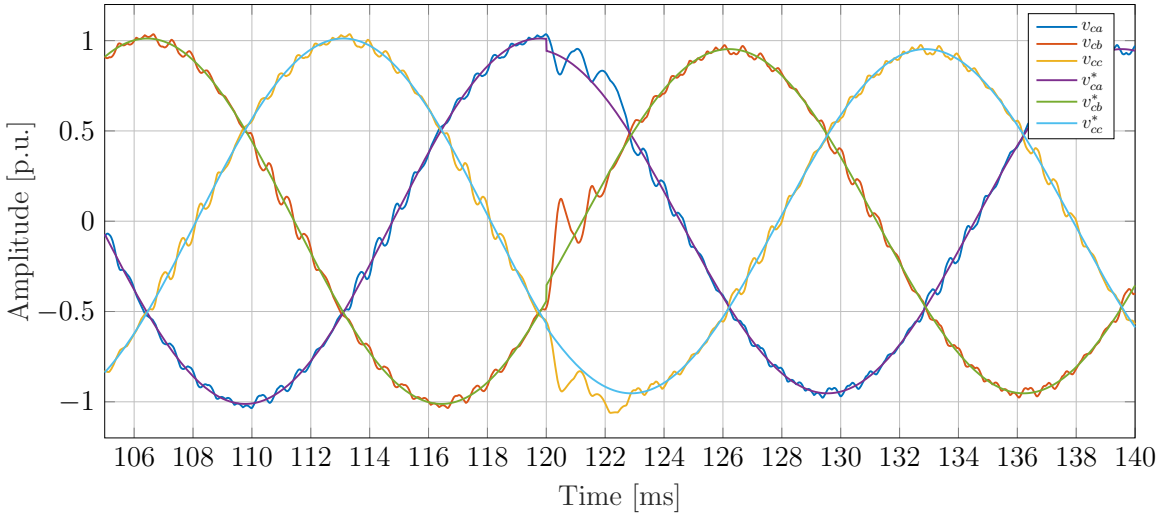
The converter- and grid-side currents are shown in Fig. 5.1a and Fig. 5.1b, respectively. The filter capacitor voltages are shown in Fig. 5.1c.



(a) Inverter currents.



(b) Grid currents.



(c) Capacitor voltages.

Figure 5.1: State variable simulation results for the proposed controller, with a step in reference at $t = 120\,\text{ms}$. The state variables along with their respective references are shown.

The modulation signals are shown in Fig. 5.2, and Fig. 5.3 shows the output of the pulse-width modulator.

The grid current spectrum is calculated from the steady-state grid currents and is shown in Fig. 5.4, with the harmonics as a percentage of the fundamental component amplitude.

The THD of the grid current at steady state is 0.66% at nominal grid current. When the reference grid current is at 0.5 PU, the THD is 1.09%. The fundamental component of the grid-current has an amplitude of 5842 A, which is close to the reference amplitude of 5844 A. The largest harmonic occurs at 1495 Hz with an amplitude of 29.97 A. The controller has a settling time of roughly 2.5 ms, as the reference signals step from 0.5 PU to the nominal value for I_g.

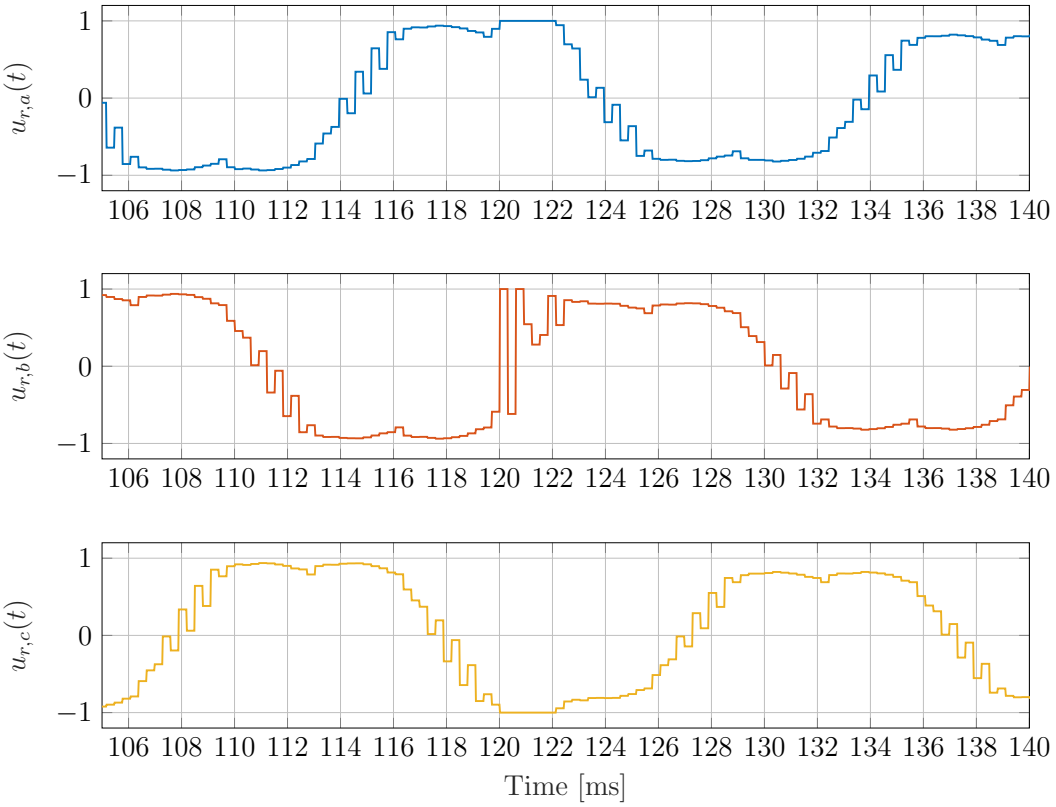


Figure 5.2: Modulating signals of proposed controller method with a step in grid current reference at $t = 120\,\text{ms}$.

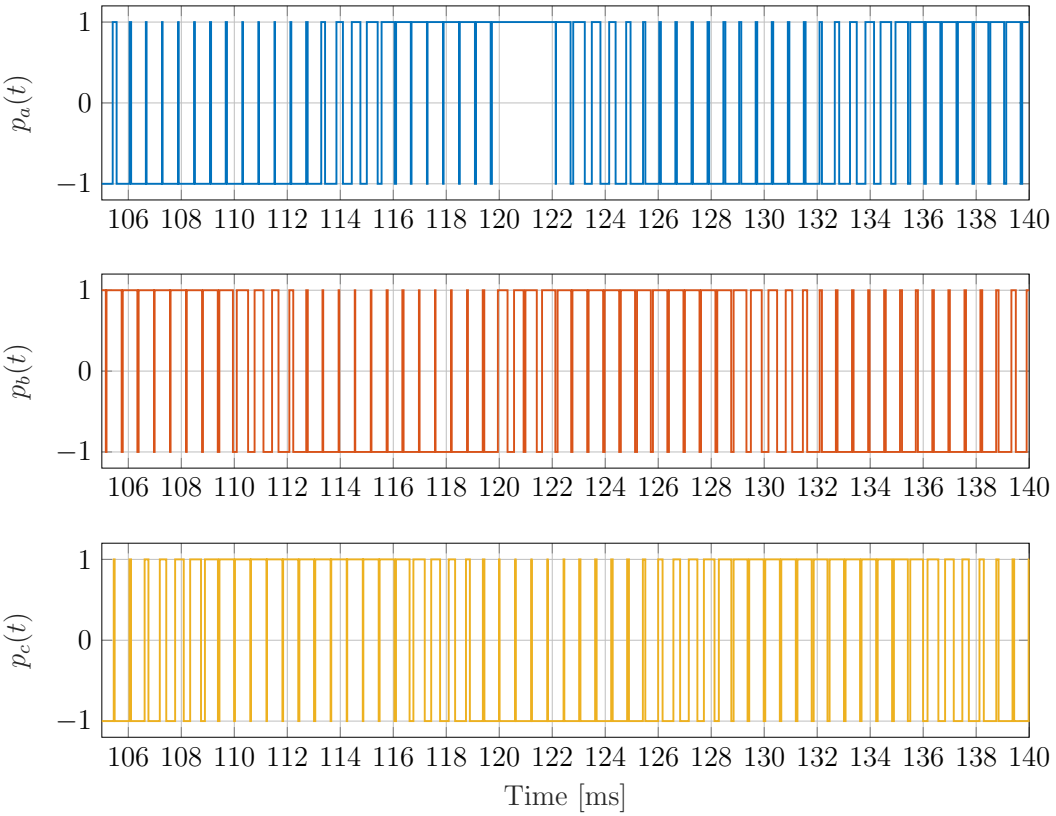


Figure 5.3: PWM output of proposed controller with a step in grid current reference at $t = 120\,\text{ms}$.

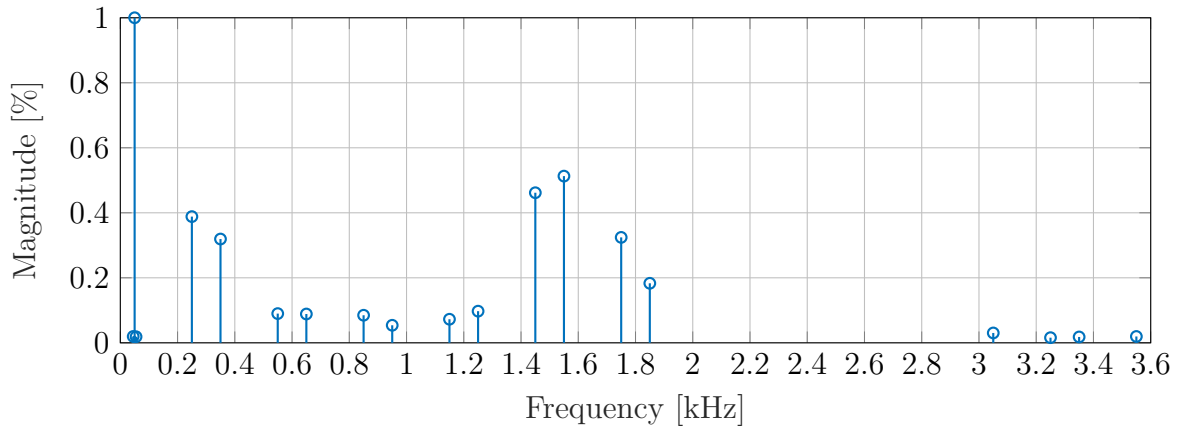


Figure 5.4: Grid current spectrum at steady-state nominal grid current.

Care must be taken with the dimension of the control problem, with regards to the implementation of the controller on an FPGA. The prediction horizon determines the size of the control problem at hand. The weight on the change in PWM reference signals λ_u is also important, since it sets the trade-off between control effort and the tracking error. Fig. 5.5 shows the relationship between the prediction horizon length and the THD of the grid current; the weight held constant at $\lambda_u = 6 \times 10^4$. In order to show the effect of the weight on the THD of the grid current, the prediction horizon is held constant at $N_p = 14$ while simulating over a range of values for λ_u, as can be seen in Fig. 5.6.

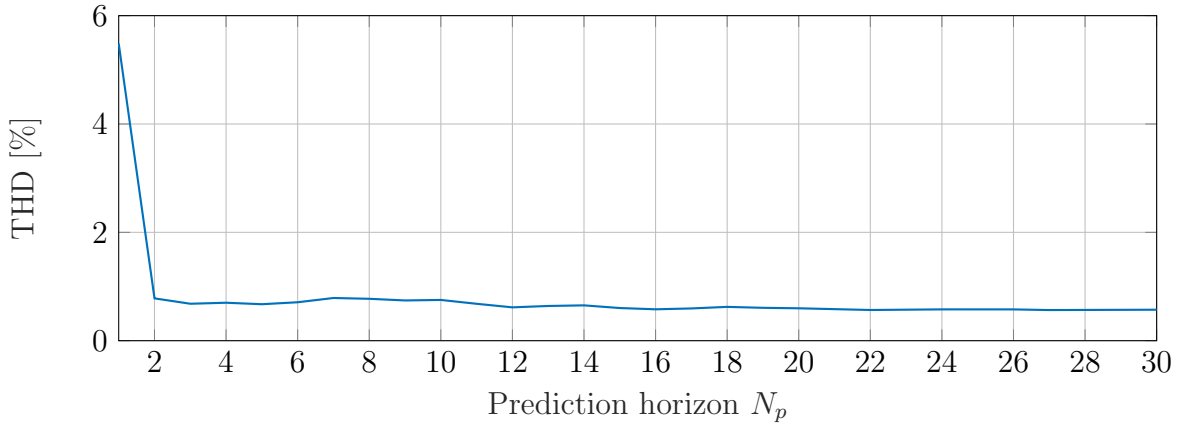


Figure 5.5: Relationship between prediction horizon and THD of grid current with $\lambda_u = 6 \times 10^4$.

It is interesting to note the large drop in THD from $N_p = 1$ to $N_p = 2$. It is also evident that from a certain value of N_p, the difference in THD between values of N_p becomes almost negligible. This means that for a controller on an FPGA device, a value of $N_p = 5$ would be ideal, since the resulting THD compares well with that of $N_p = 14$. A shorter horizon also means less multiplications are needed, since the dimension of the control problem decreases.

As expected, the THD rises with the increase in λ_u, as the controller becomes less reactive. In order to determine the optimal combination of prediction horizon N_p length and weight λ_u value, multiple simulations were run over a range of prediction horizons as well as

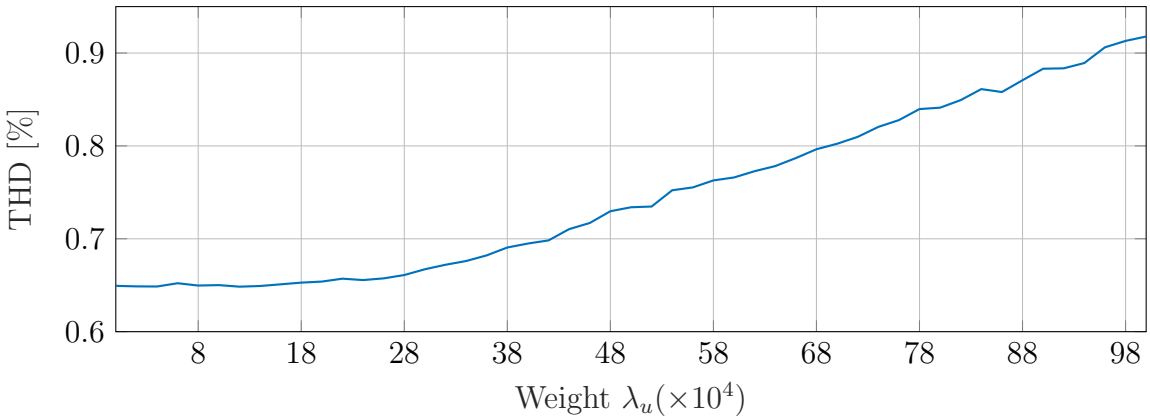


Figure 5.6: Relationship between PWM reference weight and THD of grid current with $N_p = 14$.

different weight values. It was found that for any combination of $1 \leq N_p \leq 4$ and $0 \leq \lambda_u \leq 1 \times 10^4$ [1], the controller was unstable and the THD was high ($\geq 1\%$). For simplicity, only the iterations of the simulation where $N_p = 5$ and $N_p = 14$ are shown in Fig. 5.7.

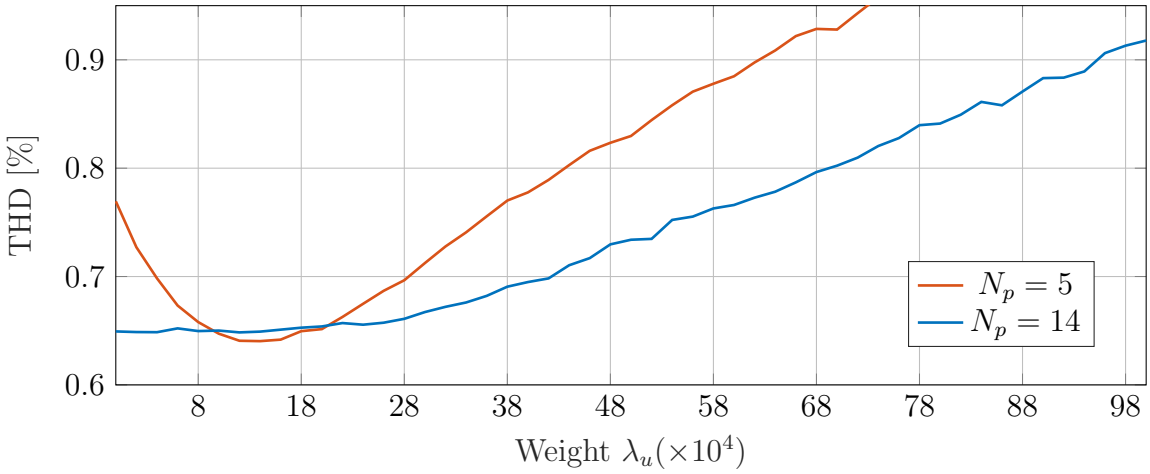
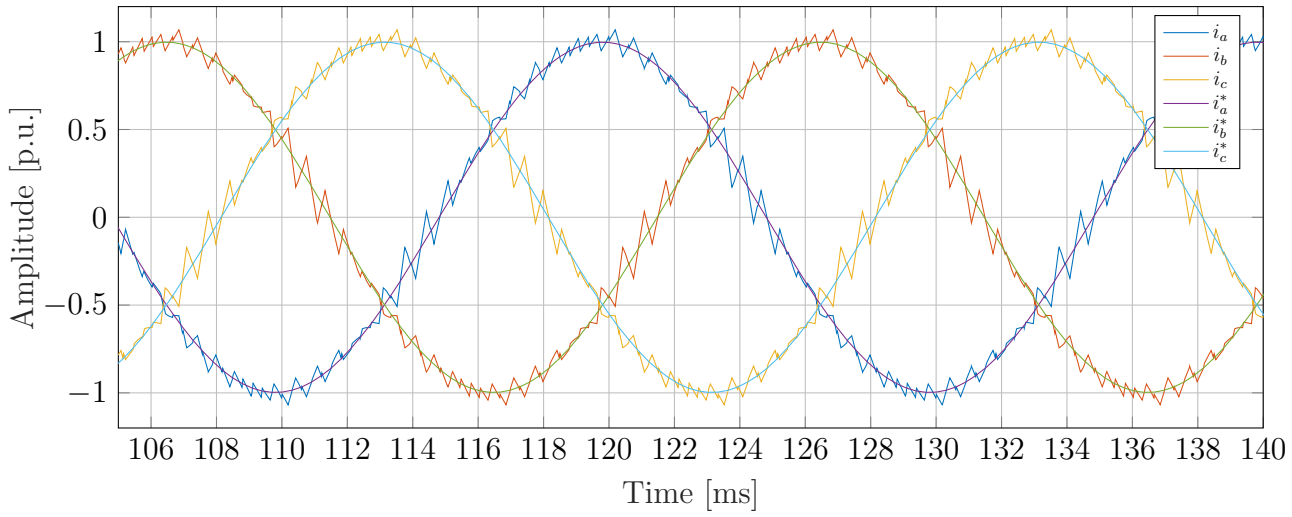


Figure 5.7: Relationship between PWM reference weight and THD with $N_p = 5$ and $N_p = 14$.

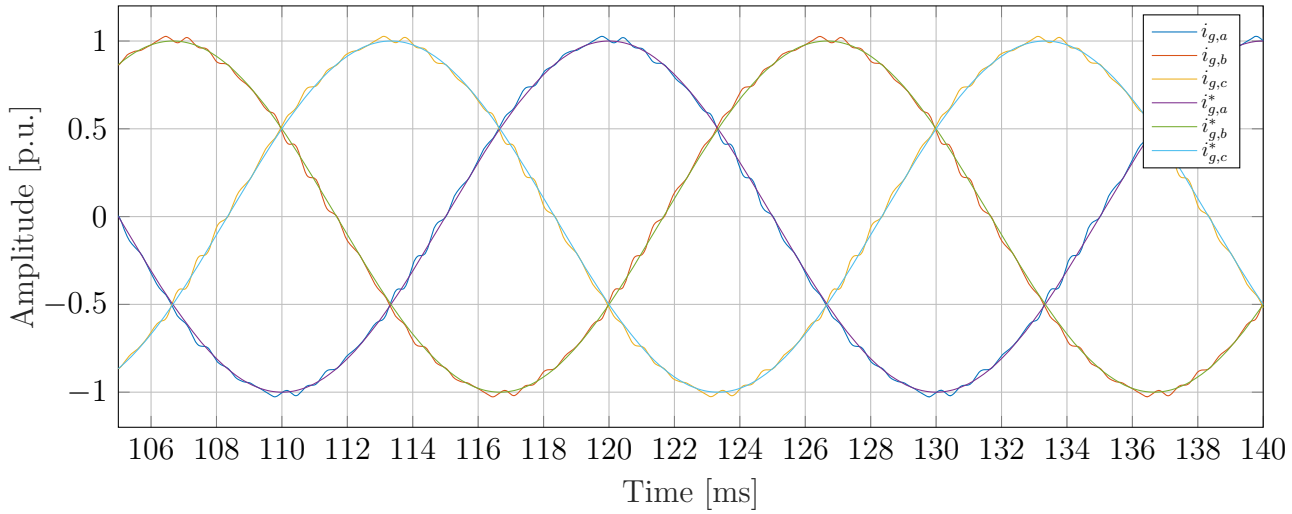
As evident in the graph, there exists a point where $N_p = 5$ and $\lambda_u = 14 \times 10^4$, where the lower prediction horizon produces a lower THD of the grid current than the case for $N_p = 14$. These parameters are, therefore, chosen for implementation on the FPGA.

In order to test the robustness of the controller with varying parameters, the grid side inductance L_g of the plant is reduced to 50% of its nominal value, from 44.38 µH to 22.2 µH. The grid current reference is kept at its nominal value. The results are compared with the appropriate reference signals and are shown in the figures below. The converter and grid currents are shown in Fig 5.8a and Fig 5.8b, respectively. The filter capacitor voltages are shown in Fig 5.8c.

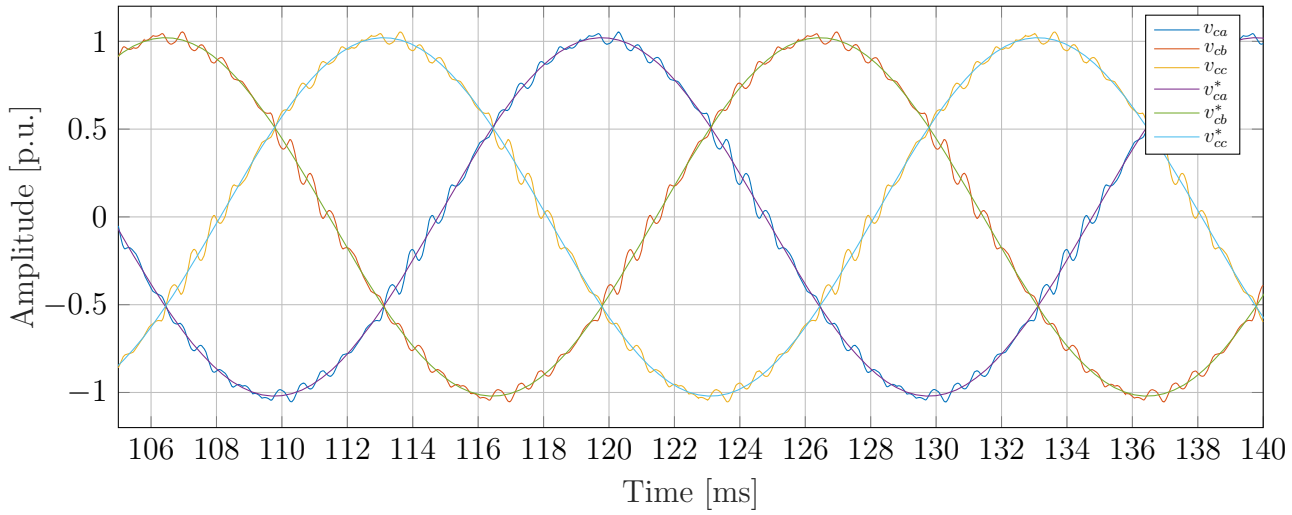
[1] Note that the value of λ_u is not given in PU. The value 1×10^4 roughly translates to 20×10^{-2} in the normalised system.



(a) Inverter currents.



(b) Grid currents.



(c) Capacitor voltages.

Figure 5.8: Simulation results with grid side inductance L_g reduced to 50% of its nominal value from 44.38 µH to 22.2 µH. The state variables along with their respective references are shown.

The THD of the grid current increased from 0.66% to 1.43%, due to the decrease in attenuation of the switching harmonics by the smaller grid inductance. The increase in THD is also due to the fact that the controller model was not updated to accommodate the change in grid inductance. The fundamental component of the grid current has an amplitude of 5866 kA, which is 0.38% larger than the nominal value.
The grid current spectrum is again calculated from the steady-state grid currents and is shown in Fig. 5.9, with the harmonics as a percentage of the fundamental component amplitude. The grid current harmonics are slightly larger, as can be expected with the decreased attenuation. The largest harmonic occurs at 1495 Hz with an amplitude of 65.24 A.

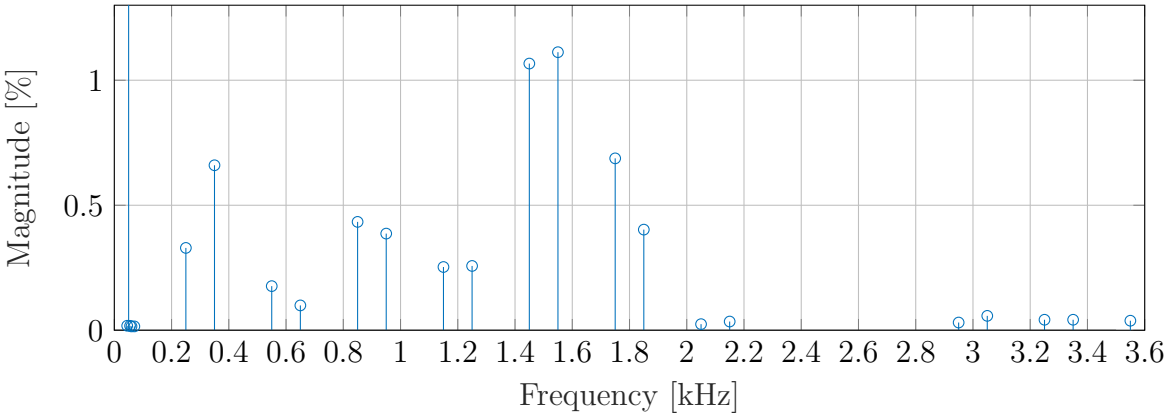


Figure 5.9: Grid current spectrum at steady-state nominal grid current with reduced L_g.

In general, the controller fares well during steady-state and transient conditions. The controller produces grid currents with a very low THD. The system also responds relatively quickly to a step in reference signal and settles within one eighth of the fundamental period. The instant right after the reference step is applied, it is evident that there is a slight delay in the controller's response. This was due to the value of the weight λ_u, which prevented the controller to react aggressively.

The results prove that the controller does not need an active damping strategy in order to dampen the resonant frequency of the *LCL*-filter. In contrast, classic PI-based controllers with an active damping loop require switching frequencies of at least three times the resonance frequency of the filter [8].

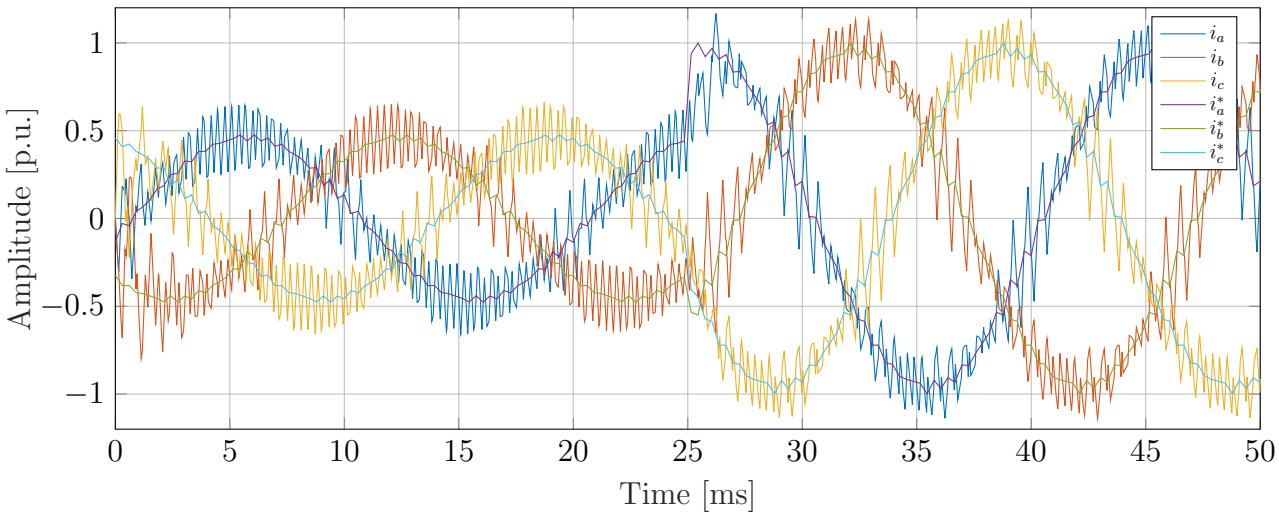
5.3 LQR Controller

The mathematical system model of the plant is realised in a Matlab-based simulation. The system parameters are given in Table 5.2. A thorough simulation of this controller can be found in [8]. This section only presents some of the simulation results of [8] for brevity.

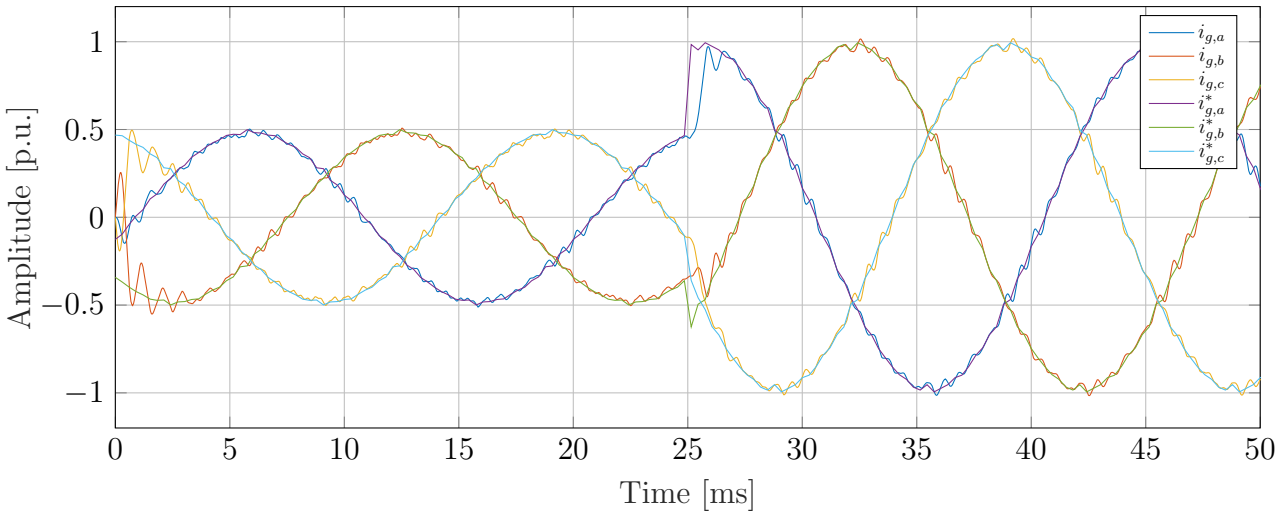
Table 5.2: System parameters for small signal LQR simulation.

Parameter	Symbol	Value
Filter inductance	L	30 µH
Grid inductance	L_g	29.19 µH
Filter capacitance	C	1.98 mF
Filter inductor ESR	R	54 mΩ
Grid inductance ESR	R_g	1.1 mΩ
Filter capacitor ESR	R_C	0.667 mΩ
DC-link voltage	V_d	1.32 kV
Grid voltage (line-to-line)	V_g	400 V (rms)
Fundamental grid frequency	f_1	50 Hz
Switching frequency	f_c	1.65 kHz
Simulation sampling interval	T_s	606.07 ns

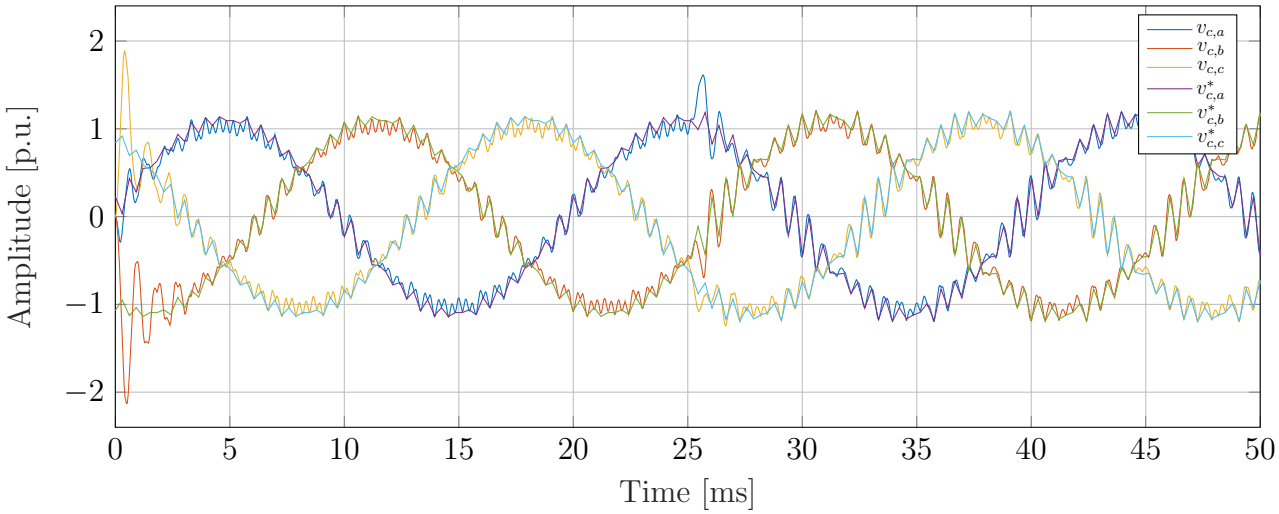
The simulation results are presented below. A step in the grid current from 0.5 PU to nominal value is introduced to the system at time $t = 25$ ms to show the transient response. Fig 5.10a and Fig 5.10b show the converter and the grid currents, respectively. Fig. 5.10c shows the filter capacitor voltages.



(a) Inverter currents.



(b) Grid currents.



(c) Capacitor voltages.

Figure 5.10: Simulation results of LQR controller with step in grid current reference at time $t = 25\,\text{ms}$. The state variables along with their respective references are shown.

It can be seen that the controller responds well to a step in reference, with a settling time of roughly 3 ms. The small-signal controller output is shown in Fig 5.11. The controller only makes small adjustments during the steady-state. At the startup and step in reference transients, the controller makes larger corrections to force the state variables towards the reference trajectories.

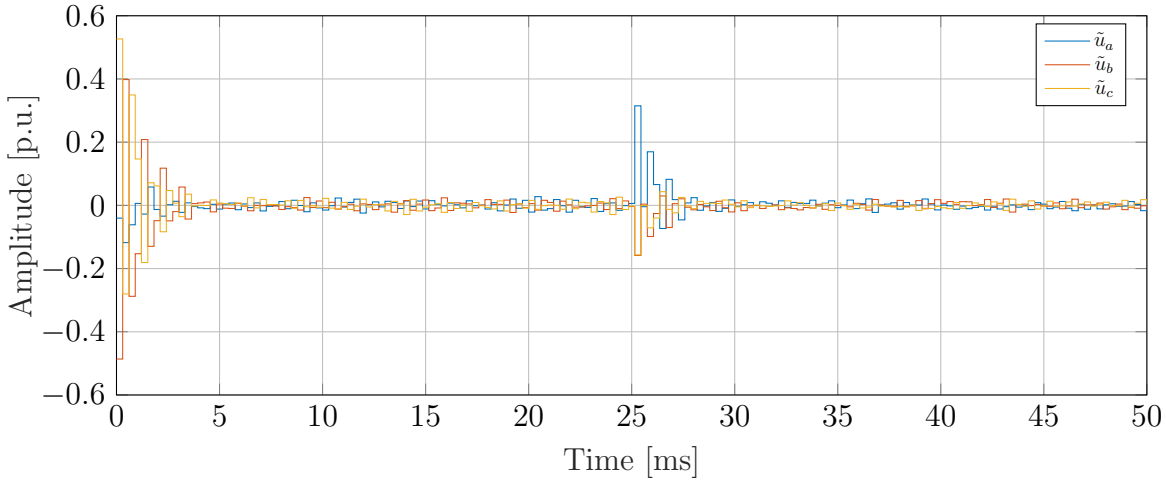


Figure 5.11: Small signal adjustments of the LQR controller with step in reference at $t = 50\,\text{ms}$.

The grid current spectrum for steady-state at nominal grid current is shown in Fig 5.12, with the harmonics as a percentage of the fundamental.

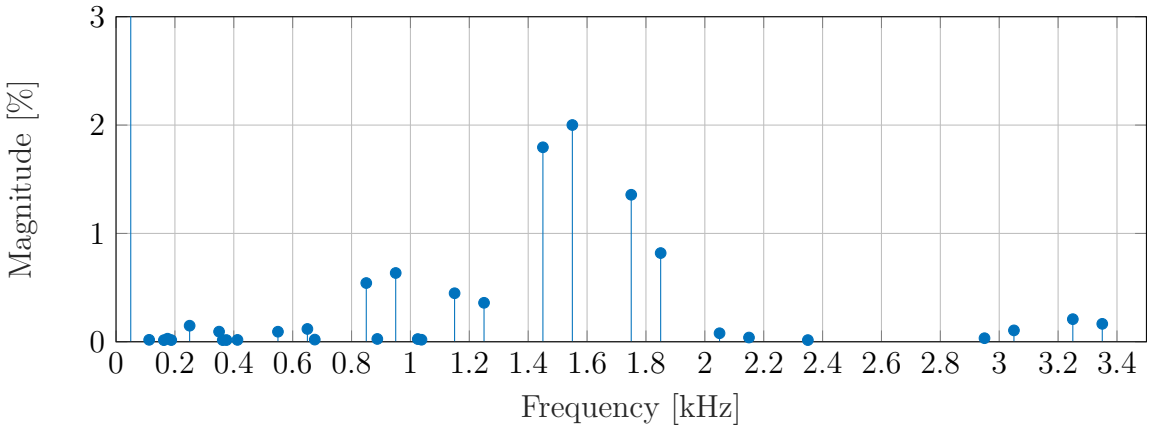


Figure 5.12: Grid current spectrum at steady-state nominal grid current for LQR controller.

Different results were acquired due to parameter changes in this thesis with regards to [8]. The largest harmonic occurs at a frequency of 1550 Hz, which is close to the switching frequency of 1650 Hz. The fundamental component is 98% of the reference amplitude of I_g^*.

As concluded in [8], the main advantage of the small-signal LQR controller is that it enables the converter to operate at less than twice the resonance frequency of the *LCL*-filter, when compared to other PI-based controllers that require a switching frequency of up to three times the resonance frequency.

5.4 Summary

Two scenarios were investigated to determine the efficacy of the proposed controller. The controller had a fast settling time when a step in grid current reference was applied. The controller had a very low THD, and proved to be effectively robust to a change in plant variation. Different combinations of prediction horizons N_p and weighting factors λ_u were tested, after which an optimal point was found and chosen for controller implementation on an FPGA. The controller also operates well at a switching frequency less than twice the resonant frequency of the filter.

The small-signal LQR controller results showed a very low THD as well as a fast transient response to a step in grid current reference. Like the proposed controller, the LQR controller is able to operate at a switching frequency less than twice the resonant frequency of the filter.

Both sets of system and control parameters of the controllers were tuned individually to achieve the lowest harmonic distortion in each case. For further research it is recommended to use the same system parameters for both, and only tune the controller parameters in each case to do a better comparison between the two techniques on a like-for-like basis. This was, however, not the case in this document and it makes it difficult to sufficiently compare the controllers.

Chapter 6

FPGA Implementation

6.1 Introduction

This chapter discusses the implementation of the proposed controller as well as the alternative LQR controller on an FPGA device. The first section starts by discussing a few preliminary concepts and tools necessary for the implementation. An introduction to VHDL is firstly given. The FPGA used for the implementation is then discussed. Finally, a few arithmetical concepts are discussed, namely how to simplify an equation for implementation in VHDL (such as $\boldsymbol{\Theta}$ in (4.14) in Section 4.3), the use of fixed-point representation versus floating-point, and lastly the concept of an adder tree. The second section explains the principle of an ideal control system with zero computational delay and discusses the timing delay compensation of a practical system. The third section gives an overview of a hardware-in-the-loop simulation, stating some of its advantages and discussing its implementation. The fourth section explains in detail the implementation of the LQR controller as well as the proposed controller in VHDL. A conclusion is given in the final section.

6.2 Preliminaries

6.2.1 VHDL

An FPGA is an integrated circuit with programmable logic and, therefore, requires a hardware description language (HDL) that can be synthesized by a synthesizer tool (Vivado, from Xilinx in this case) into physical hardware logic on the FPGA. The abbreviation VHDL stands for VHSIC (Very High Speed Integrated Circuits) Hardware Description Language. The advantages of VHDL concerning this project are listed below [31]:

- VHDL supports concurrent logic. This means that multiple hardware tasks are able to run independently in parallel. This can increase the speed of a design, when compared to sequential logic.
- In addition to parallelism, VHDL also supports sequential logic in the form of processes. With processes, the developer is able to implement algorithms with the same type of building blocks, e.g. `if`-statements and `for`-loops, as other typical programming languages.

- VHDL provides the developer the ability to define specific types as well as packages with functions and procedures, allowing for the language to be extended.
- The language supports good design principles with constructs for abstraction, modularity, and hiding information. Good design principles reduces the coding mistakes and helps with the readability of the code.

The disadvantages of VHDL are listed below [31]:

- VHDL is a complex language and difficult to master. In order to write effective VHDL code, a deeper understanding of the underlying models and concepts are necessary as it is not always apparent in the code. The parallel nature of the hardware can also be a difficult principle to grasp initially.
- VHDL is a low-level language, most apparent by the lack of object-oriented programming.
- The code in VHDL is verbose, with similar constructs in other programming languages requiring much less typing.

In short, VHDL is a powerful tool that can be utilized best when the developer has a deeper understanding of the execution model and its concepts. When considering the practical implementation of MPC, it is apparent from the advantages mentioned above that VHDL is well suited for the task.

6.2.2 FPGA Description

As mentioned in Section 3.9.2, the Digilent ZedBoard XC7Z020 development board is used for practical implementation of the two proposed controllers [32]. The ZedBoard contains a Xilinx XC7Z020-1CLG484C Zynq-7000 AP SoC architecture, which consists of a dual-core ARM Cortex A9 processing system and a Xilinx FPGA with capabilities close to that of the Xilinx Artix-7 XC7A100T FPGA. Only the FPGA component of the Zynq-7000 SoC will be used for the implementation of the controllers.

The programmable logic[1] contains 53200 look-up-tables (LUTs), 106400 flip-flops, 4.9 MB RAM (or 140 blocks of 36 kB), and 220 DSP slices. Each DSP slice has an 18×25 two's compliment signed multiplier as well as a 48-bit adder/accumulator. Both the multiplier and adder are "capable of operating up to 741 MHz", given that the design is adequately optimised [30].

The design of a controller in VHDL requires intimate knowledge of the physical capabilities of the FPGA on which it is implemented. For example, the delay induced by a synthesized chain of look-up-tables, adders, multipliers, flip-flops, etc. cannot exceed the period of the internal clock of the FPGA. The design also cannot consist of a magnitude of calculations, since the number of embedded DSP units is limited. The Xilinx Vivado integrated design environment (IDE) provides a useful project summary that displays the resource usage on the FPGA for the current design. The IDE also provides a detailed implemented timing report where time constraint violations are shown. These tools help the developer to maintain an effective design.

[1] Programmable logic refers to the FPGA component of the Zynq-7000 chip.

There are powerful FPGAs, such as the Intel Stratix series, where design optimisation is almost unnecessary, however, the goal of this thesis is to implement both controllers on a cost-effective low-budget device, such as the Xilinx XC7Z020.

6.2.3 Arithmetic in VHDL

The multiplication of fixed-point numbers is the operator that quickly uses up most of the DSP slices and must, therefore, be carefully used. As mentioned in Section 3.9.2, there exists a trade-off between the amount of resources used and the amount of time required for each calculation. In order to spare resources, more clock cycles may be used for each calculation. When time is in short supply, then more resources can be used to do calculations simultaneously. Lets say, e.g., that a certain function has to compute the dot-product of two vectors V_1 and V_2, with $V_1, V_2 \in \mathbb{R}^{10}$, and has to do so within 10 clock cycles. The developer can choose to do a multiplication at each clock cycle, therefore, reusing one multiplier each time and fitting into the time constraint of 10 cycles. The second option would be to do all 10 multiplications in a single clock cycle, thus saving time but at the cost of 10 DSP slices.

This section provides a few techniques that can be used to optimise the design of a controller in VHDL, such as simplifying the equations by using the simplification of $\boldsymbol{\Theta}$ in (4.14) as an example, using adequate fixed-point numbers and explaining the concept of an adder tree.

Simplification of Equations

Let's revisit the equation for $\boldsymbol{\Theta}$ in Section 4.3. For convenience the equation is given again as

$$\boldsymbol{\Theta}^\top(k) = 2\Big\{[\boldsymbol{\Gamma}\boldsymbol{x}(k) - \boldsymbol{X}^*(k)]^\top \boldsymbol{Q}_c\boldsymbol{\Upsilon} + \boldsymbol{V}_g^\top(k)\boldsymbol{\Omega}^\top\boldsymbol{Q}_c\boldsymbol{\Upsilon} + \lambda_u[\boldsymbol{E}\boldsymbol{K}\boldsymbol{u}(k-1)]^\top\boldsymbol{S}\Big\}\,. \quad (6.1)$$

It can be shown that the calculation of $\boldsymbol{\Theta}$ for a prediction horizon N_p, without any manipulation, will require $56N_p + 118N_p^2$ multiplications. Even when ignoring all the accompanying additions, the total number of multiplications will make it impossible to implement on a device with only 220 DSP slices, since the gradient projection method will require much more computational power than when calculating $\boldsymbol{\Theta}$. Let's say, e.g. that $N_p = 5$, the synthesised design of the process that calculates $\boldsymbol{\Theta}$ requires 50 DSP slices, and the system clock runs at 20 MHz. The calculation will then require 65 clock cycles, or 3.25 µs to execute. In the example, quite a large amount of multipliers are required and a relatively large time delay is incurred. Clearly, a different approach has to be taken with the on-line calculation of the controller optimisation stage.

When examining $\boldsymbol{\Theta}$, it can be seen that only $\boldsymbol{x}(k)$, $\boldsymbol{X}^*(k)$, $\boldsymbol{V}_g^\top(k)$, and $\boldsymbol{u}(k-1)$ change with every sample. This means that the other matrices can be multiplied off-line and loaded into look-up tables of the FPGA. The equation for $\boldsymbol{\Theta}$ can be rearranged as

$$\boldsymbol{\Theta}^\top(k) = 2\Big\{\boldsymbol{x}^\top(k)\boldsymbol{\Gamma}^\top\boldsymbol{Q}_c\boldsymbol{\Upsilon} - \boldsymbol{X}^{*T}(k)\boldsymbol{Q_c}\boldsymbol{M} + \boldsymbol{V}_g^\top(k)\boldsymbol{\Omega}^\top\boldsymbol{Q}_c\boldsymbol{\Upsilon} + \lambda_u\boldsymbol{u}^\top(k-1)[\boldsymbol{K}^\top\boldsymbol{E}^\top\boldsymbol{S}]\Big\} \quad (6.2)$$

which leads to

$$\boldsymbol{\Theta}^{\top}(k) = 2\Big\{\boldsymbol{x}^{\top}(k)\boldsymbol{R}_1 - \boldsymbol{X}^{*T}(k)\boldsymbol{R}_2 + \boldsymbol{V}_g^{\top}(k)\boldsymbol{R}_3 + \boldsymbol{u}^{\top}(k-1)\boldsymbol{R}_4\Big\} \ , \tag{6.3}$$

with

$$\boldsymbol{R}_1 = \boldsymbol{\Gamma}^{\top}\boldsymbol{Q}_c\boldsymbol{\Upsilon} \ , \ \boldsymbol{R}_2 = \boldsymbol{Q}_c\boldsymbol{\Upsilon} \ , \ \boldsymbol{R}_3 = \boldsymbol{\Omega}^{\top}\boldsymbol{Q}_c\boldsymbol{\Upsilon} \ , \text{ and } \boldsymbol{R}_4 = \lambda_u\boldsymbol{K}^{\top}\boldsymbol{E}^{\top}\boldsymbol{S} \ . \tag{6.4}$$

The static matrices $\boldsymbol{R}_1$, $\boldsymbol{R}_2$, $\boldsymbol{R}_3$, and $\boldsymbol{R}_4$ can now be calculated off-line with the help of MATLAB. The new equation for $\boldsymbol{\Theta}$ requires only $108N_p$ multiplications, which is a reduction of 83.3% in the number of multiplications necessary. The entries of the matrices are then loaded one per line into a text file stored in the directory of the VHDL project, from where the Xilinx Vivado synthesizer reads the values during synthesis. These values are then loaded into the on-board RAM of the FPGA during programming, and are then available to use during on-line calculations.

Fixed Point Numbers

There are two options for representing numeric values with fractional parts, namely floating-point and fixed-point numbers. Floating-point representations are easier to work with for developers, because they can handle a dynamic range and do not require the developer to specify the size beforehand. Floating-point, however, is much more complex and computationally taxing than fixed-point representation and very difficult to implement on an FPGA because of the lack of a floating-point unit (FPU). In this regard, the fixed-point representation is more suited to the application. The developer has to be aware of the quantisation error induced by the fixed-point representation, as the accuracy of each number is determined by the amount of fractional bits assigned thereto.

Adder Trees

With multiplication already discussed, the focus is turned to addition in this section. When the need arises to sum multiple real-valued numbers in VHDL (such as with matrix multiplications), a developer might be tempted to do the calculation as

$$a = b_1 + b_2 + b_3 + +b_4 + \cdots + b_n \ , \tag{6.5}$$

or maybe even create a summing variable a, where each value of b_n is added to the value of a within a `for`-loop. These approaches seem sound in theory, and could be implemented optimally with an advanced synthesizer. The Vivado synthesizer does not, however, synthesize such a calculation optimally. The above equation would typically be synthesized using a chain of look-up tables and adders. The structure of the synthesized model is depicted in Fig. 6.1 with a string of cascaded summation blocks.
The chain of summations has a length of $k = N - 1$ when N values has to be summed. When the amount of additions is large enough, the chain structure causes large timing constraint violations when implemented in a single clock cycle. One solution to this problem is known as an adder tree, where pairs of numbers are added in parallel and

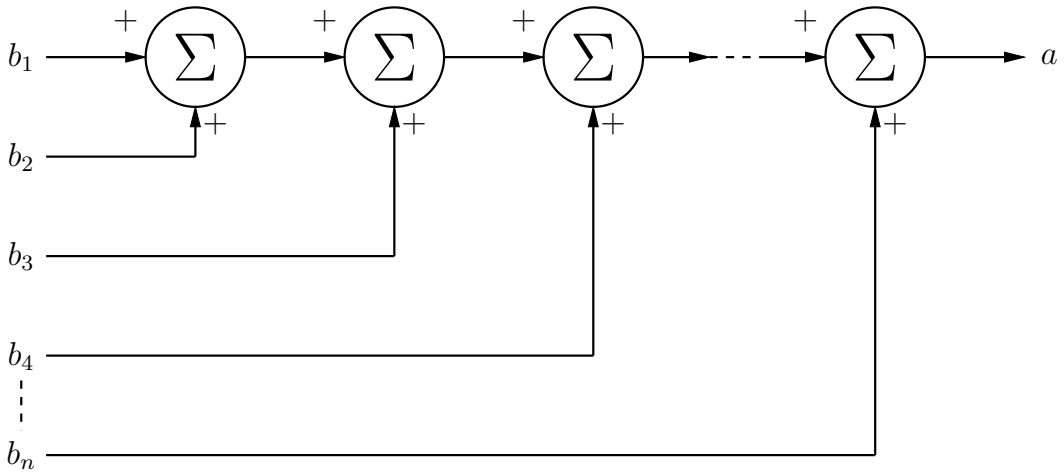


Figure 6.1: Depiction of a cascaded chain of summations.

stored. The pairs of results are then added until the final answer is reached. The principle can be easily explained with Fig. 6.2

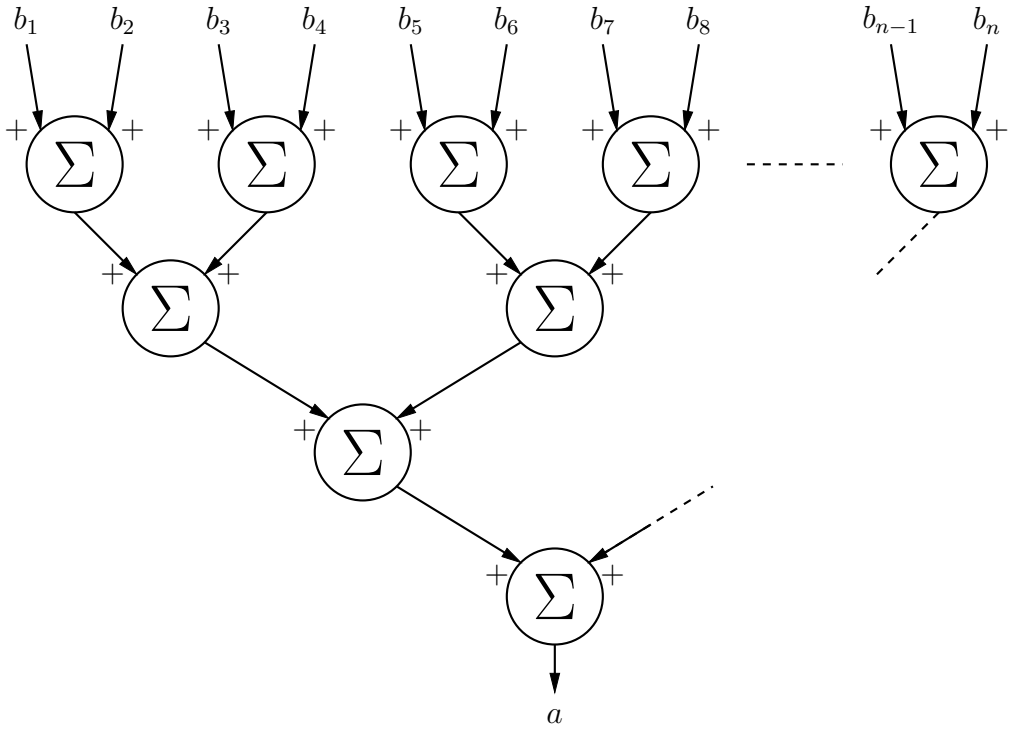


Figure 6.2: Depiction of an adder tree.

When N values are summed, the amount of levels k in the adder tree is calculated as

$$k = \lceil \log_2(N) \rceil \ , \tag{6.6}$$

where $\log_2$ denotes the base two logarithm and $\lceil \epsilon \rceil$ denotes the value of ϵ rounded up to the nearest integer. Lets say, e.g., that $N = 1e4$. The adder tree chain would be $k = 14$ levels long, whereas the cascaded chain would be $k = 9999$ summations long. The adder tree clearly decreases the length of the cascaded chain of summations.

6.2.4 Delay Compensation

When a simulation is run in a Matlab script, the state variables are sampled, the optimal reference signals are calculated and then applied instantaneously to the pulse-width modulator at every sampling instant. The ideal scenario is depicted in Fig. 6.3 only for the a-phase, with $f_c(t)$ denoting the carrier signal and with $u_{opt,a}(t)$ and $p_a(t)$ denoting the optimal a-phase reference signal and the resulting pulse-width modulated waveform, respectively.

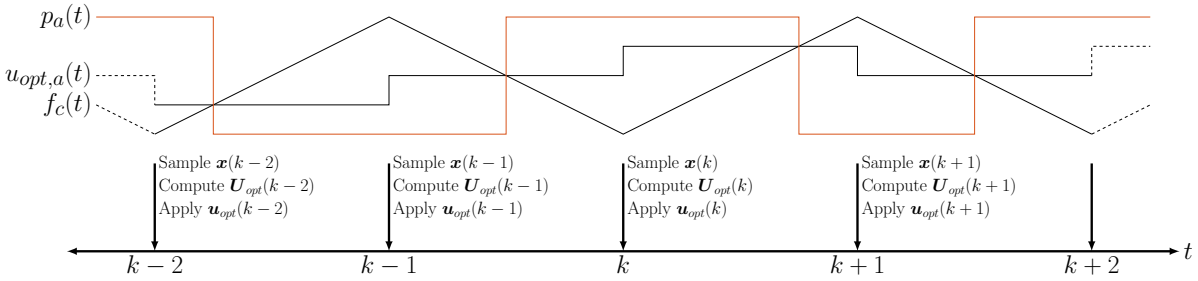


Figure 6.3: Depiction of an ideal scenario where there is no computational delay.

In a practical system, however, the ideal case is not possible, since each calculation requires a certain amount of time to execute. The use of an analogue-to-digital converter (ADC) also introduces delays to the control system. For the gradient projection method, the optimisation stage consists of calculating $\boldsymbol{\Theta}$ and then implementing the gradient projection method, which is then iterated N_f times. If the amount of iterations is fixed, then the computational time delay t_c is also fixed.

When the next optimal reference $\boldsymbol{u}_{opt}(k)$ is calculated, it can only be applied at the following sampling instant $k+1$. The controller assumes that the reference will be applied at k, not $k+1$ and certainly not in between the two instants. The computational time delays are depicted in Fig. 6.4. The top grey rectangles indicate the shifted PWM signal which is caused by the delayed application of $\boldsymbol{u}_{opt}$.

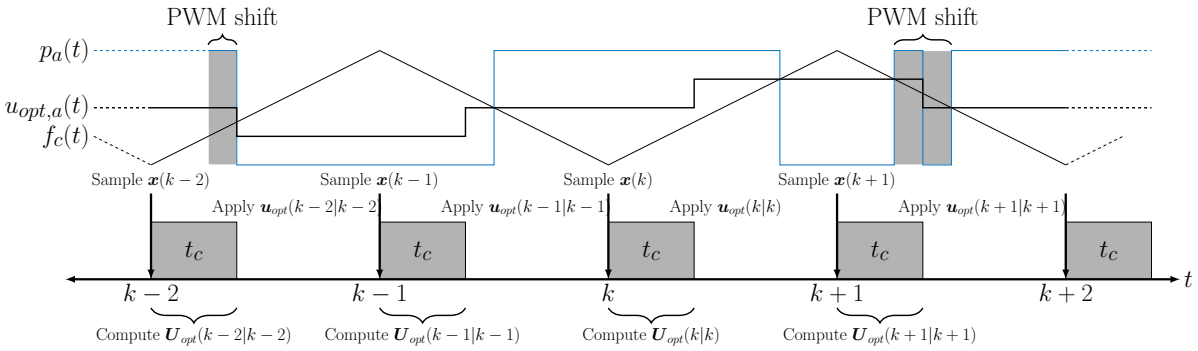


Figure 6.4: Depiction of a practical scenario where a computational delay is introduced without any delay time compensation.

In order to compensate for the computational delay, the next sample of the state vector can be predicted using the state-space model of the system. Using this predicted value,

the optimal references at the following instant can be calculated in advance. If the state vector is sampled at $k+m$, then the following $k+m+1$ sample can be calculated by using the discrete-time state space model, discretised with the controller sampling period T_c. The equation is given as

$$\boldsymbol{x}(k+1|k) = \boldsymbol{A}_{T_c}\boldsymbol{x}(k|k-1) + \boldsymbol{B}_{T_c}\boldsymbol{p}(k) + \boldsymbol{\mathcal{V}}_{T_c}\boldsymbol{v}_g(k|k-1) \tag{6.7}$$

where the matrices $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{\mathcal{V}}$ are calculated as

$$\boldsymbol{A}_{T_c} = \boldsymbol{e}^{\boldsymbol{F}T_c} \tag{6.8}$$
$$\boldsymbol{B}_{T_c} = -\boldsymbol{F}^{-1}(\boldsymbol{I}_{n_x} - \boldsymbol{A}_{T_c})\boldsymbol{G} \tag{6.9}$$
$$\boldsymbol{\mathcal{V}}_{T_c} = -\boldsymbol{F}^{-1}(\boldsymbol{I}_{n_x} - \boldsymbol{A}_{T_c})\boldsymbol{P}\ . \tag{6.10}$$

The notation $\boldsymbol{u}_{opt}(k+m|k+m-1)$ indicates the optimal reference vector at $k+m$ as calculated at $k+m-1$. The predicted sample can then be used to calculate the optimal sequence of reference vectors. The optimisation problem can be written as

$$\begin{aligned}\boldsymbol{U}_{opt}(k+1|k) = \arg\underset{\boldsymbol{U}(k+1|)}{\text{minimise}}\ & J(\boldsymbol{x}(k+1|k), \boldsymbol{x}^*(k+1|k), \boldsymbol{U}(k+1|k)) \\ \text{s.t. } & \boldsymbol{U} \in \mathbb{U}\ ,\end{aligned} \tag{6.11}$$

The first optimal reference vector $\boldsymbol{u}_{opt}(k+1|k)$ of $\boldsymbol{U}_{opt}(k+1|k)$ is then applied to the pulse-width modulator.
The system with delay time compensation is illustrated in Fig. 6.5.

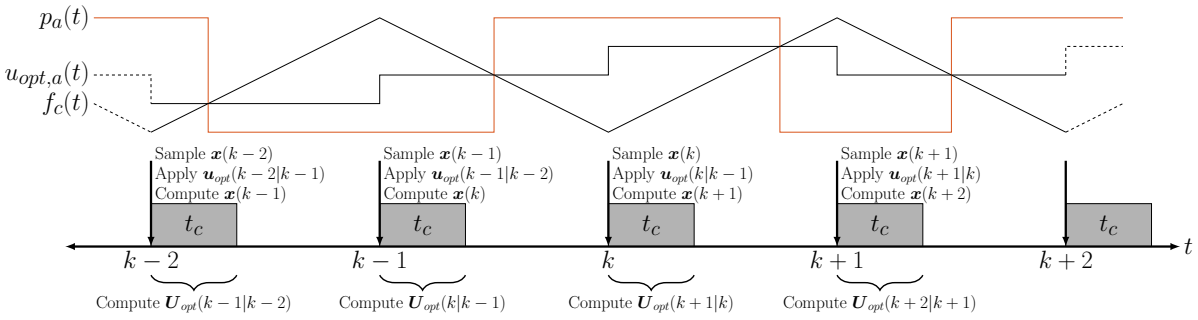


Figure 6.5: Depiction of a practical scenario where the controller compensates for the computational time delay.

6.2.5 Hardware-in-the-loop Simulation

The controllers can be tested by using what is known as a hardware-in-the-loop, or HiL, simulation. In the industry, HiL systems are often used to test the feasibility of a control system without having to build the full practical set up, or without having to debug while connected to the power grid. The HiL device receives the analogue and/or digital signals from the controller, calculates the next state variables of the simulated model and then outputs the updated states to the controller. The HiL has the following advantages:

- The HiL provides a safe testing environment, e.g. in the case of fault currents in a converter, the signals are only digital and will not cause a converter to combust or electric shock to the operator.
- Testing on a HiL is much cheaper than building a practical set up from scratch.
- The HiL system can provide more information on signals that would typically require a magnitude of measurement devices in a practical system.
- The design of a HiL typically requires much effort, but will save a lot of time during the implementation phase of a controller.

When implementing a controller on an FPGA, an extra process can be created that serves as an 'internal' HiL. A global array of logic vectors is created which stores the state variables at the current instant. These states are fed to the controller, which in turn calculates the optimal switching signals. The switch values are then fed to the HiL process, which updates the state variables, thus, closing the loop.

6.3 Controller Implementation in VHDL

This section explains in detail the implementation of the proposed LQR controller as well as the proposed controller in VHDL. The simpler LQR method (in terms of VHDL implementation) is firstly discussed. The following section walks the reader through the steps of creating the more complex gradient projection method, by giving an overview of the flow of the controller and explaining the entire optimisation stage, that includes the calculation of $\boldsymbol{\Theta}$ as well as the gradient projection method.

6.3.1 LQR Method

One of the advantages that the LQR controller has over the proposed controller, is that it is much easier to implement on-line in an FPGA. The complex calculation of the steady-state reference signals can entirely be done off-line with the help of MATLAB. These steady-state reference signals are then loaded into the RAM of the FPGA for on-line use. The only computational effort required for the LQR controller is the calculation of the product of the error signal and the feedback gain, which is then added to the pre-computed steady-state reference signal to perform corrections.

It has to be stated that the simplicity of the LQR controller depends on the voltage from the grid as well as the desired grid current. For an accurate controller, the reference signals must be updated on-the-fly as the grid voltage changes, resulting in dynamically changing steady-state PWM reference trajectories. These reference trajectories can, however, be calculated off-line for a variety of models with different phase angles and modulation indices. The off-line trajectories are then stored in look-up tables on the FPGA device. By making use of zero-crossing detection, and adjusting the position of the references, a result can be achieved close to that of the on-the-fly reference calculation method, as mentioned in [8]. Only the off-line implementation is considered in this document.

The next few sections discuss the processes required for the LQR implementation. Every process is driven by the system clock with frequency f_{clk}, in order to synchronise the

processes. Sub-clocks are generated from f_{clk} inside a process using counters. There are two different sub-clocks in this design. The first executes at the controller sampling frequency f_c, and is used only for the controller. The second sub-clock executes at the HiL simulation sampling frequency and drives the carrier, HiL simulation, and PWM generating processes.

LQR Algorithm Process

The flow of the LQR controller process is shown in Algorithm 2.

Algorithm 2 Small signal LQR Algorithm

function LQR METHOD($\boldsymbol{U}^*$,$\boldsymbol{x}(k)$,$\boldsymbol{X}^*$,$\boldsymbol{v}_g(k)$)
  if $v_{g,a}(k) = 0$ **and** $v_{g,a}(k-1) < 0$ **then** ▷ Check a-phase zero-crossing.
    $\ell \leftarrow 0$ ▷ Reset reference sample counter.
  end if
  $\tilde{\boldsymbol{x}}(k) = \boldsymbol{X}^*(\ell) - \boldsymbol{x}(k)$ ▷ Calculate error.
  $\tilde{\boldsymbol{u}}(k) = \boldsymbol{K_f}\tilde{\boldsymbol{x}}(k)$ ▷ Add feedback gain.
  $\tilde{\boldsymbol{u}}_{abc}(k) = \boldsymbol{K}^{-1}\tilde{\boldsymbol{u}}(k)$ ▷ Transform to abc.
  $\boldsymbol{u}(k) = \boldsymbol{U}^*(\ell) + \tilde{\boldsymbol{u}}_{abc}(k)$ ▷ Update PWM references.
  if $\ell < (N_u - 1)$ **then** ▷ Check for end of reference samples.
    $\ell = \ell + 1$ ▷ Increment reference sample counter.
  else
    $\ell = 0$ ▷ Reset reference sample counter.
  end if
end function

The process uses the signals $\boldsymbol{U}^*$, $\boldsymbol{x}(k)$, $\boldsymbol{X}^*$ and $\boldsymbol{v}_g(k)$ as 'inputs'. The symbols $\boldsymbol{U}^*$ and $\boldsymbol{X}^*$ refer to the pre-calculated PWM reference trajectories in abc and state reference trajectories in $\alpha\beta$, respectively, over a fundamental period T_1, where

$$\boldsymbol{U}^* = \begin{bmatrix}\boldsymbol{u}^*(0) & \boldsymbol{u}^*(1) & \cdots & \boldsymbol{u}^*(\ell) & \cdots & \boldsymbol{u}^*(N_u-1)\end{bmatrix} \text{ and} \tag{6.12}$$

$$\boldsymbol{X}^* = \begin{bmatrix}\boldsymbol{x}^*(0) & \boldsymbol{x}^*(1) & \cdots & \boldsymbol{x}^*(\ell) & \cdots & \boldsymbol{x}^*(N_u-1)\end{bmatrix} . \tag{6.13}$$

The variable ℓ keeps track of the references sample instant. The symbols $\boldsymbol{x}(k)$ and $\boldsymbol{v}_g(k)$ denote the current sample of the state vector and the grid voltage, respectively. The symbol $\boldsymbol{K}_f$ refers to the feedback gain matrix, whereas $\boldsymbol{K}^{-1}$ denotes the inverse Clarke transform matrix.

Carrier Process

A second process generates the triangular carrier signal by using a counter and a `counting_up` boolean variable. Once the carrier reaches the value $+1$, the boolean `counting_up` = `false` and the counter starts to count down, decrementing the value of the carrier. Once the carrier reaches -1, the boolean `counting_up` = `true` and the counter counts up again, incrementing the value of the carrier. The size of the counter N_c is the ratio between the system clock f_{clk} and the carrier frequency f_c, i.e. $N_c = \frac{f_{clk}}{f_c}$. The constant value c, with which the carrier variable is incremented and decremented at each time instant, can be found by dividing the size of the counter N_c by twice the amplitude of the carrier (which is 1), i.e. $c = \frac{N_c}{2}$.

PWM Generator Process

The third process simply compares each phase in the reference vector $\boldsymbol{u}^*(k)$ with the carrier signal and generates PWM signals, accordingly. This process executes at the same frequency at which the carrier signal is updated.

6.3.2 Gradient Projection Method

This section describes the different processes that form the proposed controller. The first section describes the process that controls the flow of every stage of the controller. The following sections discuss in detail the approach that was taken with the calculation of $\boldsymbol{\Theta}$ and the gradient projection method implementation.

Controller Flow

In order to keep track of all the processes that makes up the proposed controller, a *logic tracker* process is created. This tracker, in essence, serves as a state machine that operates the controller. Each component of the controller is allocated a certain number of clock cycles to execute, depending on the complexity of the computation. This means that the *logic tracker* keeps track of time by monitoring the number of clock cycles that each component requires. The controller is sampled at twice the frequency of the carrier. The time between controller sampling instants can be expressed as $T_{ctr} = \frac{T_c}{2}$, where T_c is the period of the carrier. The computational time of the controller t_c has to be less than T_{ctr}.

Fig. 6.6 depicts the timeline that the controller follows. The top time axis shows the controller sampling intervals as well as the carrier signal, with the computational time t_c of the entire controller indicated as dark-grey rectangles. The second time axis shows a zoomed-in view of the timeline of controller processes between sampling instants, each indicated with a $\boldsymbol{\mathcal{P}}_x$. The third time axis shows a zoomed in view of the optimisation stage process $\boldsymbol{\mathcal{P}}_5$, which consists out of two underlying processes. The first, $\boldsymbol{\mathcal{P}}_{5,1}$, depicts the step in the steepest direction of the gradient projection method. The second, $\boldsymbol{\mathcal{P}}_{5,2}$, depicts the projection step of the gradient projection method. The final process $\boldsymbol{\mathcal{P}}_6$ simply 'primes' the optimal references for application at the next sampling interval.

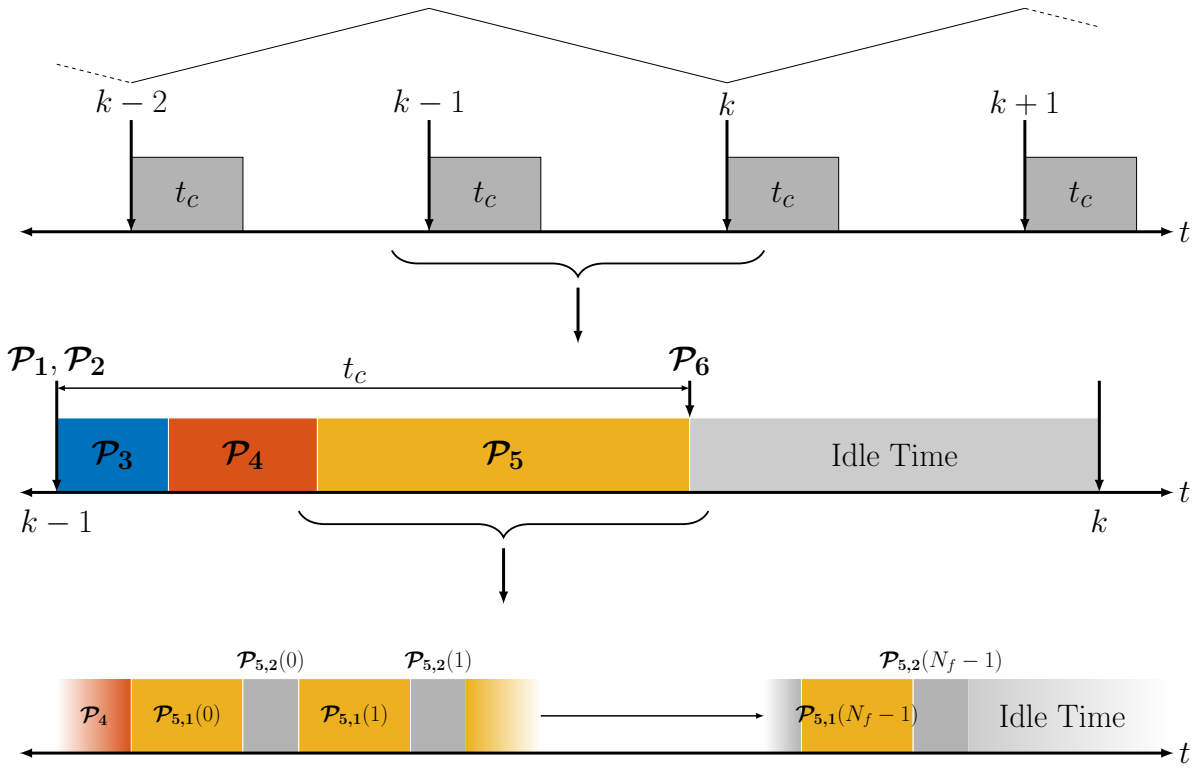


Figure 6.6: Depiction of the zoomed-in timeline that the gradient projection method VHDL implementation follows between controller sampling intervals.

A brief description of the processes in Fig. 6.6 is given below:

- $\mathcal{P}_1$: This process updates the optimal reference signal $\boldsymbol{u}_{opt}(k|k-1)$ at the beginning of the controller sampling period, as calculated in the previous sampling interval. No calculation is required, since the signals only exchange values, therefore, requiring only a single clock cycle.

- $\mathcal{P}_2$: This process calls a function that updates the *prediction horison tracker*. At every controller sample, the calculation of $\boldsymbol{\Theta}$ requires the vectors $\boldsymbol{X}^*(k)$ and $\boldsymbol{V}_g(k)$, which consists of values from the current sample k up to the prediction horizon $k+N_p-1$. The *prediction horison tracker* is a vector of indices that track the progression of the two input vectors $\boldsymbol{X}^*(k)$ and $\boldsymbol{V}_g(k)$. This process also requires only one clock cycle and is executed at the same time as $\mathcal{P}_1$.

- $\mathcal{P}_3$: This process calls a *state update* function, that uses the current sample k of the state vector and predicts the the state vector at the next sampling instant $k+1$. The *state update* function calculates each state variable in the vector separately at one per clock cycle, therefore, requiring n_x clock cycles to execute. The predicted state vector is then used in the following process, which calculates $\boldsymbol{\Theta}$.

- $\mathcal{P}_4$: This process calls a function that calculates the value of $\boldsymbol{\Theta}$. The function receives the predicted value of the next sample $\boldsymbol{x}(k+1)$, the state reference vector $\boldsymbol{X}^*(k)$ and the grid voltage vector $\boldsymbol{V}_g(k)$ over the prediction horizon, and the current optimal reference vector $\boldsymbol{u}_{opt}(k)$ as inputs. The function calculates each one of the $2N_p$ entries of $\boldsymbol{\Theta}$ separately at one per clock cycle. The whole process,

therefore, requires $2N_p$ clock cycles. The calculation of $\boldsymbol{\Theta}$ is discussed in more detail in a later subsection.

- $\boldsymbol{\mathcal{P}}_5$: This process serves as a second *logic tracker*, which tracks the progress of the gradient projection optimisation, which is iterated N_f times to produce the optimal reference vector $\boldsymbol{U}_{opt}(k)$ over the prediction horizon. The notation $\boldsymbol{\mathcal{P}}_{5,1}(n)$ is used, where n indicates the n'th iteration. The two underlying processes are discussed in more detail in a later subsection.
  - $\boldsymbol{\mathcal{P}}_{5,1}(n)$: The first of the two underlying processes uses the gradient of the cost function to calculate a single step in the steepest direction, using the value from the previous iteration as starting point. Each entry of $\boldsymbol{U}_n(k)$ is calculated separately at one per clock cycle, therefore, requiring $2N_p$ clock cycles for a single step.
  - $\boldsymbol{\mathcal{P}}_{5,2}(n)$: The second underlying process takes the value of $\boldsymbol{U}_n(k)$, and projects it onto the box constraints (see Fig 3.12). The values of $\boldsymbol{U}_n(k)$ are then updated in pairs at one per clock cycle, therefore, requiring N_p clock cycles.
- $\boldsymbol{\mathcal{P}}_6$: The final process of the controller uses the first entry $\boldsymbol{u}_{opt}(k)$ of the newly calculated $\boldsymbol{U}_{opt}(k)$ to 'prime' the optimal PWM reference signals for application at the following sampling instant. This process only exchanges signal values, therefore, requiring only a single clock cycle to execute.

When the amount of clock cycles of every process is added up, the total controller time t_c can be calculated as

$$t_c = t_{\mathcal{P}_{1\&2}} + t_{\mathcal{P}_3} + t_{\mathcal{P}_4} + t_{\mathcal{P}_5} + t_{\mathcal{P}_6} \tag{6.14}$$

$$= T_{clk}[1 + n_x + 2N_p + N_f(2N_p + N_p) + 1] \tag{6.15}$$

$$= T_{clk}[N_p(3N_f + 2) + n_x + 2] \tag{6.16}$$

The controller executes every $T_c = 312.5\,\mu s$. If the number of gradient projection iterations is set to $N_f = 50$, the prediction horizon is set to $N_p = 5$, the number of state variables is $n_x = 6$, and a system clock with frequency $f_{clk} = 20\,\text{MHz}$ is used, then the total computational time of the controller is $t_c = 38.4\,\mu s$. This means that there is more than enough time for the controller to execute before the following sampling instant.

Calculation of $\boldsymbol{\Theta}$ in VHDL

The first part of the optimisation stage is the calculation of $\boldsymbol{\Theta}$, since it forms part of the equation that describes the gradient of the cost function J at time step k. As mentioned in Section 6.2.3, calculating all entries of $\boldsymbol{\Theta}$ at once will require a large amount of the FPGA resources. In addition to the large amount of resources used, the computational strain on the FPGA will cause timing constraint violations, which in turn cause the process to run slower than preferred. It has been shown in Section 6.2.3 that with the simplification of $\boldsymbol{\Theta}$, the resource usage can be reduced.

In order to achieve an even higher efficiency with regards to resource usage, the calculation of each entry of $\boldsymbol{\Theta}$ can be done sequentially over $2N_p$ clock cycles. A function calculates the i'th entry of $\boldsymbol{\Theta}$ and only exchanges the coefficients at every clock cycle. This method allows the FPGA to recycle multipliers.

Recall (6.3) in Section 6.2.3, which gives the reduced form of $\boldsymbol{\Theta}$. Given the four reduced system matrices, $\boldsymbol{R}_1$,$\boldsymbol{R}_2$,$\boldsymbol{R}_3$, and $\boldsymbol{R}_4$, the i'th entry of $\boldsymbol{\Theta}$ can be written as

$$\begin{aligned}\Theta_i = &\left(\sum_{j=0}^{n_x-1} x_j \boldsymbol{R}_1(i,j)\right) + \left(\sum_{j=0}^{n_x N_p-1} X_j^* \boldsymbol{R}_2(i,j)\right) + \left(\sum_{j=0}^{3N_p-1} V_{g,j} \boldsymbol{R}_3(i,j)\right) \\ &+ \left(\sum_{j=0}^{n_u-1} u_j \boldsymbol{R}_4(i,j)\right) ,\end{aligned} \tag{6.17}$$

where the notation $\boldsymbol{R}(i,j)$ indicates the entry of matrix $\boldsymbol{R}$ with row i and column j. The $k-1$ notation of u is dropped for simplicity. Each of the four summations is carried out using an adder tree to adhere to timing constraints.

The total amount of multipliers required for the i'th entry in $\boldsymbol{\Theta}$ can be calculated by adding together all the upper limits of the four terms, i.e.

$$\#\text{ Multipliers} = n_x + n_x N_p + 3N_p + n_u \ . \tag{6.18}$$

For the system with $n_x = 6$ state variables, horizon length $N_p = 5$, and output vector $n_u = 3$, the amount of multipliers required is equal to 54. Given that the FPGA in use has 220 multipliers, the total required for the calculation of $\boldsymbol{\Theta}$ is rather small. This provides more than enough resources for the implementation of the gradient projection method.

Gradient Projection Method in VHDL

As mentioned earlier, the gradient projection method consists out of two stages. The first stage takes an unconstrained step for the optimiser $\boldsymbol{U}$ in the steepest direction, given as

$$\boldsymbol{U}_{z+1} = \boldsymbol{U}_z - \frac{1}{L}(\boldsymbol{H}\boldsymbol{U}_z + \boldsymbol{\Theta}) \ , \tag{6.19}$$

where z indicates the z'th iteration of $\boldsymbol{U}$. The step also requires the step size $\frac{1}{L}$, the off-line calculated Hessian $\boldsymbol{H}$, and the previously calculated vector $\boldsymbol{\Theta}$. To reduce the amount of resources required for calculating the step, the same strategies are employed as with $\boldsymbol{\Theta}$. Firstly, (6.19) can be simplified by rearranging all the matrix products that can be calculated off-line. The new, reduced step equation is written as

$$\boldsymbol{U}_{z+1} = \boldsymbol{U}_z - \tilde{\boldsymbol{H}}\boldsymbol{U}_z - \frac{1}{L}\boldsymbol{\Theta} \ , \tag{6.20}$$

with $\tilde{\boldsymbol{H}} = \frac{1}{L}\boldsymbol{H}$ now calculated off-line as well. The new equation requires $2N_p$ less multiplications. The computational burden can further be reduced by calculating each entry of $\boldsymbol{U}_{z+1}$ separately at one per clock cycle. The i'th entry in $\boldsymbol{U}_{z+1}$ can be written as

$$U_{i,z+1} = U_{i,z} - \left(\sum_{j=0}^{2N_p-1} \tilde{\boldsymbol{H}}(i,j) U_{i,z} \right) - \frac{1}{L}\Theta_i \ . \tag{6.21}$$

The step process requires only $2N_p$ multiplications per entry i, and subsequently, only $2N_p$ clock cycles to execute.

The second stage of the gradient projection method is the projection step. Algorithm 3 describes the method in which the box constraints are applied to the unconstrained iteration of $\boldsymbol{U}_{z+1}$. For convenience, the notation $z+1$ is dropped in the algorithm. The projected vector and output of the algorithm is denoted as $\hat{\boldsymbol{U}}$.

Algorithm 3 Gradient Projection Method

function PROJECTION($\boldsymbol{U}$)
  for $i = 0 : N_p - 1$ **do** ▷ For prediction horizon N_p.
    $\boldsymbol{u}_{abc} = \boldsymbol{K}^{-1}\boldsymbol{U}^{\top}(2i : 2i+1)$ ▷ Entries used in pairs.
    $\tilde{\boldsymbol{u}}_{abc} = \boldsymbol{u}_{abc} - \frac{1}{2}[\min(\boldsymbol{u}_{abc}) + \max(\boldsymbol{u}_{abc})]$ ▷ Add common-mode term.
    for $j = \{a, b, c\}$ **do**
      $\hat{u}_j = \min(\max(-1, \tilde{u}_j,)1)$ ▷ Round each phase off.
    end for
    $\hat{\boldsymbol{U}}^{\top}(2i : 2i+1) = \boldsymbol{K}\hat{\boldsymbol{u}}_{abc}$ ▷ Transform to $\alpha\beta$ and update.
  end for
end function

Firstly, the $\alpha\beta$ pair entries in $\boldsymbol{U}$ are transformed to *abc* format using the inverse Clarke transform $\boldsymbol{K}^{-1}$. A common-mode term is then added to the signals to establish space-vector centering. Each phase is then projected back to the "box", i.e. rounded off to either -1 or 1. The projected values are then transformed back to $\alpha\beta$ format and the corresponding entries in $\boldsymbol{U}$ are updated.

Since the amount of multipliers required for Algorithm 3 was relatively small (6 multiplications for each transformation, thus, a total of 12 multipliers is required for the entire algorithm), no reduction of the equations was done. The algorithm does the projection for one $\boldsymbol{u}$ in $\boldsymbol{U}$ at every clock cycle, therefore, requiring N_p clock cycles to execute.

Optimisation Stage Algorithm

Now that all the underlying components of the optimisation stage have been defined, Algorithm 4 describes the full optimisation stage, where the new PWM reference $\boldsymbol{u}(k)$ is the output.

Algorithm 4 Optimisation Stage

function OPTIMISE($\boldsymbol{x}(k)$,$\boldsymbol{X}^*(k)$,$\boldsymbol{V}_g(k)$,$\boldsymbol{u}(k-1)$)
  for $i = 0 : 2N_p - 1$ **do** ▷ First calculate all entries of $\boldsymbol{\Theta}$
    compute Θ_i
  end for
  for $z = 0 : N_f - 1$ **do** ▷ Iterate the gradient projection method N_f times.
    for $q = 0 : 2N_p - 1$ **do** ▷ Step in the steepest direction.
      $U_{q,z+1} = U_{q,z} - \left(\sum_{k=0}^{2N_p-1} \tilde{\boldsymbol{H}}(q,k) U_{q,z} \right) - \frac{1}{L}\Theta_q$
    end for
    $\boldsymbol{U}_{z+1}$ = PROJECTION($\boldsymbol{U}_{z+1}$)
  end for
  $\boldsymbol{u}(k) = \boldsymbol{U}_{1:2}(k)$
end function

By adding up all the required multipliers for the optimisation stage, and using the same parameters as with the calculation of the resources required for $\boldsymbol{\Theta}$, an estimate of the required resource usage can be obtained. The total amounts to 76 DSP slices (multipliers). This number is also confirmed by the project summary in the Vivado IDE, which also states that the timing requirements are met.

6.4 Summary

This chapter provided insight on the implementation of a model predictive controller in VHDL. The chapter started off by giving the reader a description of VHDL with some advantages and disadvantages with regards to the intended design. The capabilities of the FPGA chosen for the design was then discussed and reasons were given for the choice of the FPGA model. A few concepts regarding arithmetic in VHDL were the discussed, namely, how to simplify equations for the use on an FPGA, the fixed-point representation, and lastly the concept of an adder tree. The reader was then given insight into the compensation for computational delay time, after which a brief description was given of a HiL simulation.

The chapter then moved on to the implementation of two controllers in VHDL. The method that was first discussed is the LQR controller from Section 3.10. The following section describes the design of the gradient projection method in VHDL in detail with the use of a timeline. It was shown that the gradient projection method can easily finish computation within the sampling interval of the controller, whilst using only 76 DSP slices and adhering to the timing constraints.

In Chapter 7, the controllers are both implemented in a HiL simulation, where it is showed that the gradient projection and the small-signal LQR algorithms function as expected.

Chapter 7

HiL Implementation and Results

7.1 Introduction

As previously mentioned, it is advantageous to implement a HiL simulation of a modulator to determine the practical feasibility thereof. A HiL simulation can be implemented on the FPGA that runs the controller, by adding another process which runs in parallel with the controller. This process receives the gating signals from the controller as well as simulated grid voltage signals, and then updates the state variables accordingly. The state signals are sampled by the controller, closing the feedback loop.
This chapter provides the reader with a brief description of the HiL simulation implementation. The following sections are divided into two parts. The first focuses on the implementation and the results of the proposed controller. The second focuses on the implementation and results of the LQR controller.
A photo of the test board (ZedBoard) and a screenshot of the Vivado IDE is shown in Appendix D.

7.2 Implementation of a HiL Simulation

The HiL simulation for both controllers are implemented in the same manner. A process, synchronised with the system clock, is called at the sampling frequency of the simulation f_s. These sampling instants are determined by a simulation tracker, which operates similarly to the *logic tracker* mentioned in the previous chapter.
At every sampling instant the process receives the gating signals from the pulse-width modulator as well as the simulated grid voltage signals. The process then calls a function that updates the state variables. The same function used to update the state variables for the HiL simulation is used for the delay compensation for the controller, thus saving resources. The function receives the state-matrices, the state variables at k, the grid voltages, and the gating signals. The function then returns the updated states. The HiL simulation, therefore, requires its own discretised model with period T_s.
The proposed controller requires 76 of the total number of 220 DSP slices on the FPGA. The LQR controller requires only 12 DSP slices. The HiL process is, therefore, designed to execute at a single clock cycle, i.e. at each clock cycle, the time of the simulation is advanced by T_s, expending some of the excess resources left on the FPGA. The state update function requires a total amount of 66 DSP slices, according to the project summary of the Vivado IDE.

7.3 Implementation of HiL for Proposed Controller

7.3.1 Overview

The full controller and HiL simulation system are depicted in the flow diagram in Fig. 7.1. The diagram shows how the HiL simulation, shown as a yellow box, fits into the whole system. The diagram shows how each of the processes described in the previous chapter ties in with each other. Each arrow connecting processes represents a signal that is shared between them.

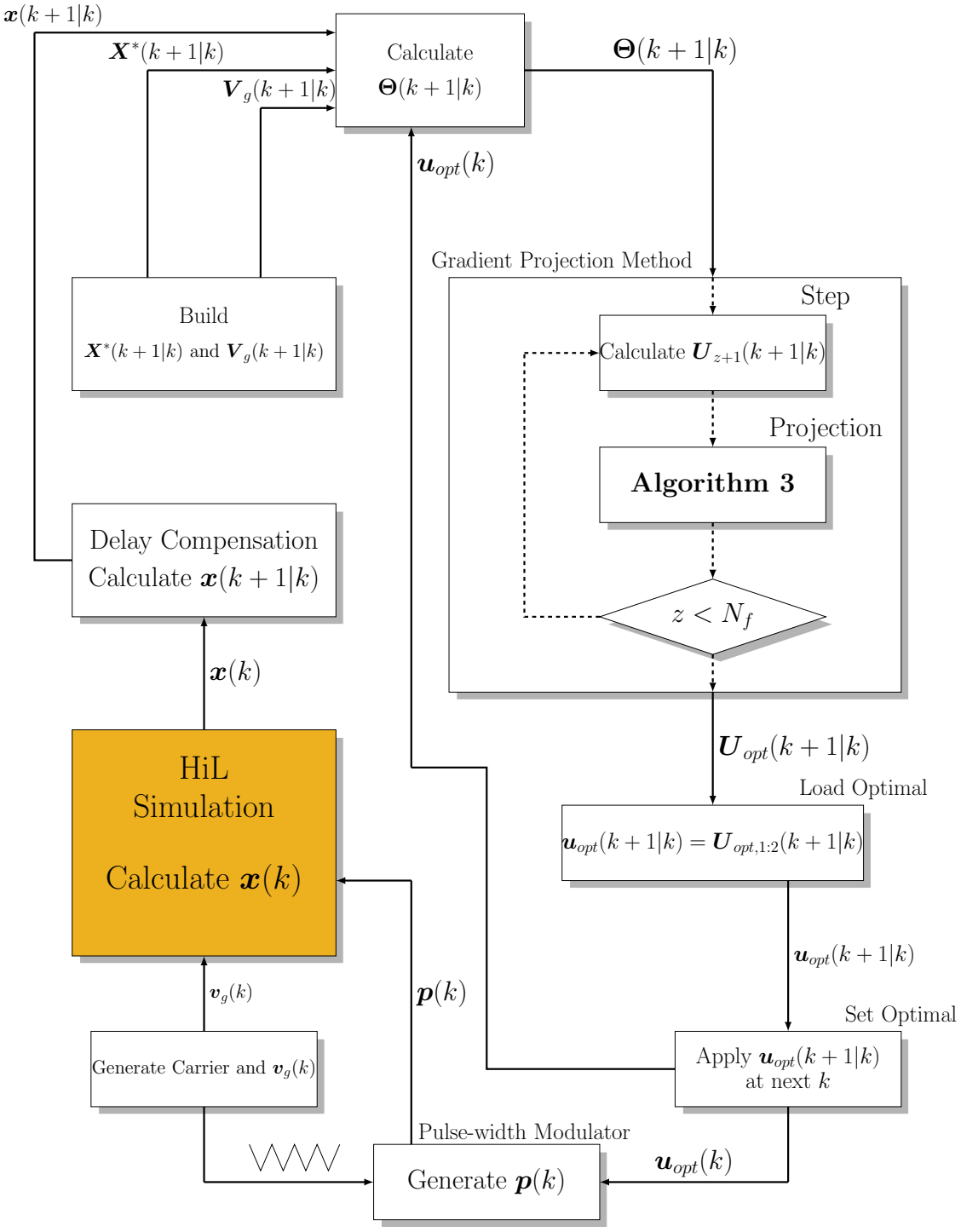


Figure 7.1: Flow diagram of proposed method controller and HiL simulation.

The control is briefly explained below:

- $\boldsymbol{\Theta}(k+1|k)$ is calculated using the previous value of the optimal reference signals $\boldsymbol{u}_{opt}(k)$, the state signals $\boldsymbol{x}(k+1|k)$ as updated by the delay compensator process, and the state reference vector $\boldsymbol{X}^*(k+1|k)$ and grid voltage vector $\boldsymbol{V}_g(k+1|k)$ as generated by the *prediction horizon tracker* function.

- The gradient projection method is then implemented using two separate processes; one calculating the step of $\boldsymbol{U}_{z+1}$ and the second implementing Algorithm 3. After all iterations are done, i.e. $z = N_f$, the optimal sequence of reference vectors $\boldsymbol{U}_{opt}(k+1|k)$ is passed on.

- The next process "Load Optimal" loads the first entry of $\boldsymbol{U}_{opt}(k+1|k)$ into a signal $\boldsymbol{u}_{opt}(k+1|k)$ in order to 'prime' the reference signals for the pulse-width modulator.

- The 'primed' signals are then applied at the next controller sampling instant by the "Set Optimal" process.

- One process constantly generates the simulated grid voltage signals $\boldsymbol{v}_g$ as well as a carrier signal.

- The pulse-width modulator compares the optimal reference signals $\boldsymbol{u}_{opt}(k)$ to the carrier, thus generating the gating signals $\boldsymbol{p}(k)$ for the simulation.

- The HiL simulation receives the gating signals $\boldsymbol{p}(k)$ and the grid voltages $\boldsymbol{v}_g(k)$ to update the state variables.

- The delay compensator uses the current state variables sample to extrapolate the values at the following controller sampling instant.

7.3.2 HiL Simulation Results for Proposed Controller

In order to test the controller implemented on the FPGA, the HiL simulation was first disconnected from the system and two fixed inputs for $\boldsymbol{x}(k)$ and $\boldsymbol{u}(k)$ was given. The test vectors that were used are given as

$$\boldsymbol{x}(k) = \begin{bmatrix} 0.1 & -0.3 & 0.05 & 0.6 & -0.19 & 0.12 \end{bmatrix}^\top$$
$$\boldsymbol{u}(k) = \begin{bmatrix} 0.3 & 0.5 & -0.2 \end{bmatrix}^\top .$$

The controller's functionality can be proved by comparing the resulting references generated by the FPGA controller to those generated in MATLAB. The comparison can be seen in Fig 7.2, where the FPGA results are plotted against MATLAB results.

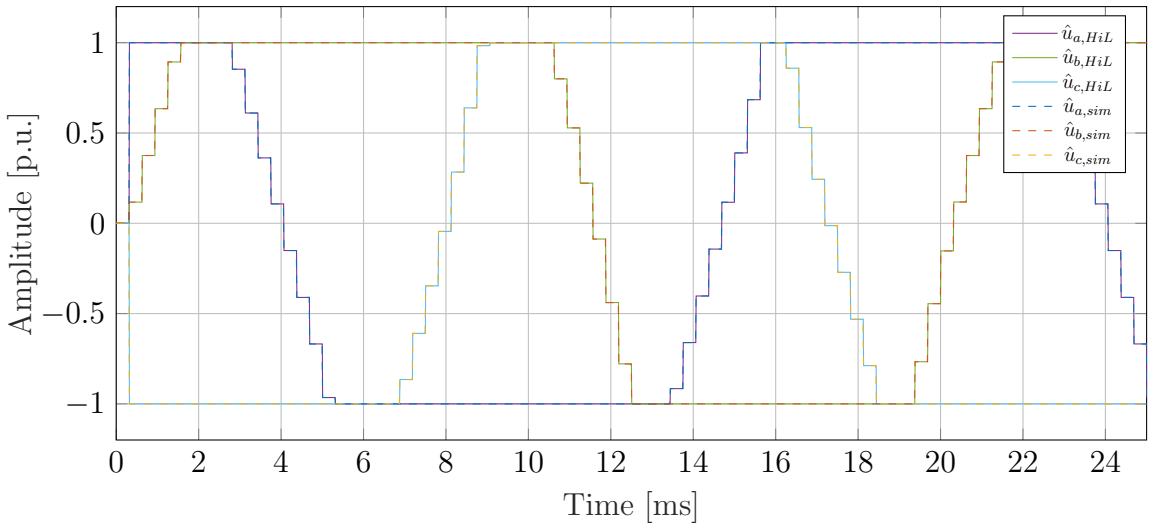


Figure 7.2: Comparison of FPGA controller generated reference signals $\boldsymbol{u}_{opt}(k)$ versus the MATLAB equivalent. Fixed values for for the inputs $\boldsymbol{x}(k)$ and $\boldsymbol{u}(k-1)$ are used.

The controller functions well as expected, with only a minor quantisation error when examining the signals up close. All the controller signals on the FPGA are 18 bit values (or words), where one acts as a sign bit, 3 bits represent the integer value, and the fractional value consists out of the remaining 14 bits.

After testing the controller, the HiL was connected to the control processes. The resulting simulated HiL signals are shown in the figures below, plotted against the MATLAB equivalent signals. Fig 7.3 shows the three-phase converter currents, where $\boldsymbol{i}_{abc,sim}$ and $\boldsymbol{i}_{abc,HiL}$ refers to the MATLAB and HiL simulated signals, respectively. Fig 7.4 shows the three-phase grid currents, where $\boldsymbol{i}_{g,abc,sim}$ and $\boldsymbol{i}_{g,abc,HiL}$ indicate the MATLAB and HiL simulated results, respectively. Fig 7.5 shows the three-phase filter capacitor voltages, where $\boldsymbol{v}_{c,abc,sim}$ denotes the MATLAB simulated signals and $\boldsymbol{v}_{c,abc,HiL}$ denotes the HiL results.

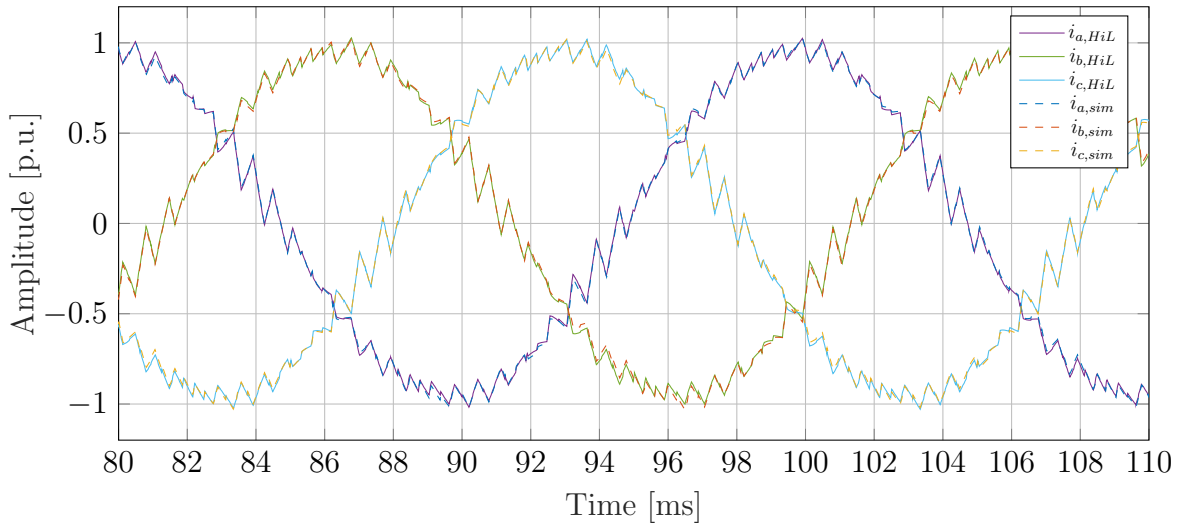


Figure 7.3: Comparison of converter current signals from HiL simulation and from MATLAB based simulation for the proposed controller.

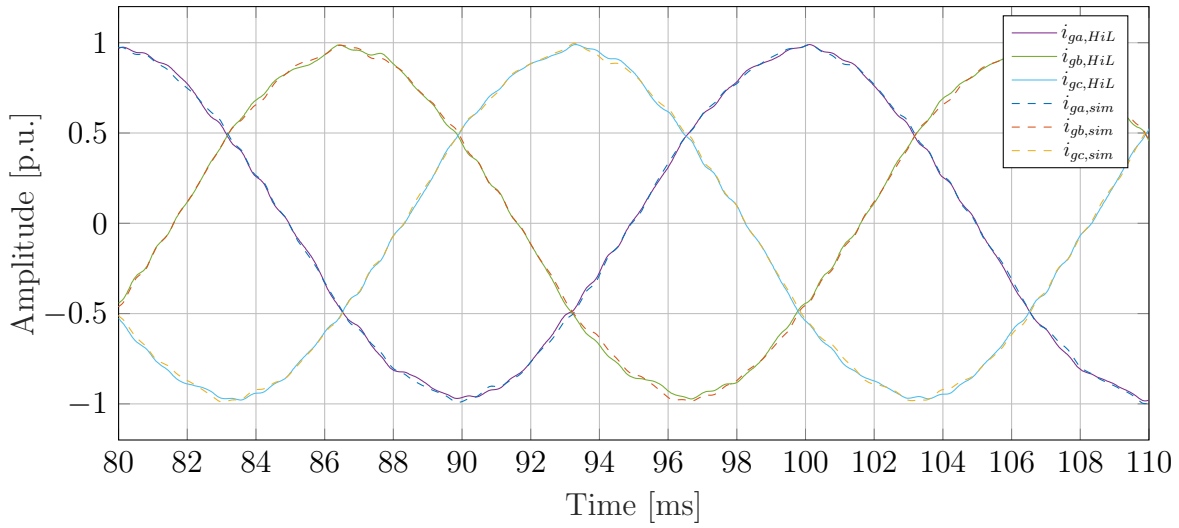


Figure 7.4: Comparison of grid current signals from HiL simulation and from MATLAB based simulation for the proposed controller.

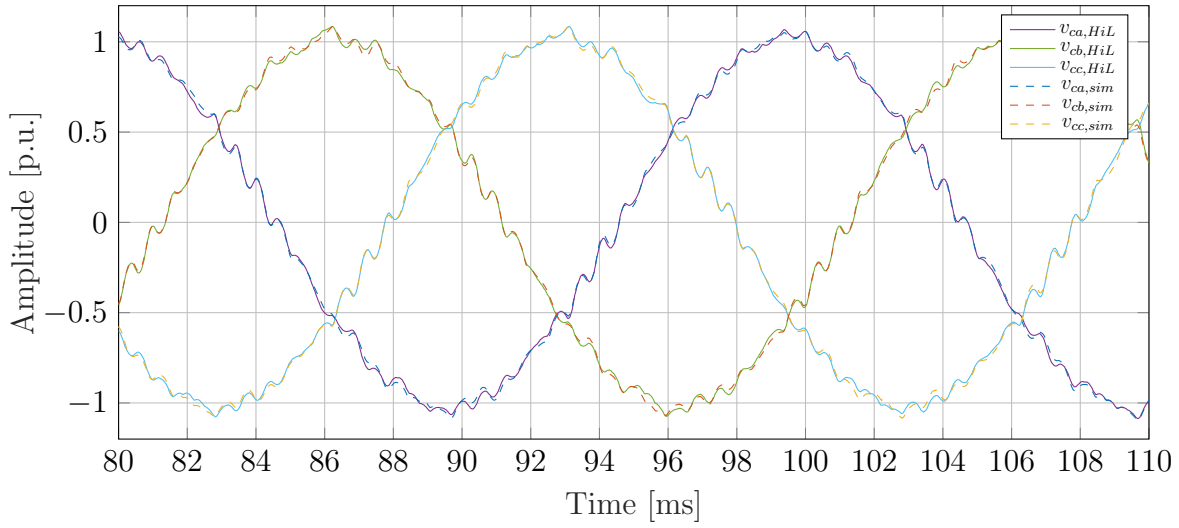


Figure 7.5: Comparison of capacitor voltage signals from HiL simulation and from MATLAB based simulation for the proposed controller.

The resulting signals of the HiL simulation compare well with the MATLAB equivalent values. There is a slight error between the HiL and Matlab results, due to a slight difference in the PWM pulses that the controller produce in each case. These PWM pulses are shown in Fig 7.6.
The difference in pulses can be ascribed to the following factors:

- The quantisation of the FPGA signals.
- The delay compensator on the FPGA calculating a next state which is slightly off.
- The carrier on the FPGA not being as accurate as the corresponding Matlab carrier signal.
- The HiL on the FPGA runs in parallel with the controller, where the states are updated sequentially in MATLAB.

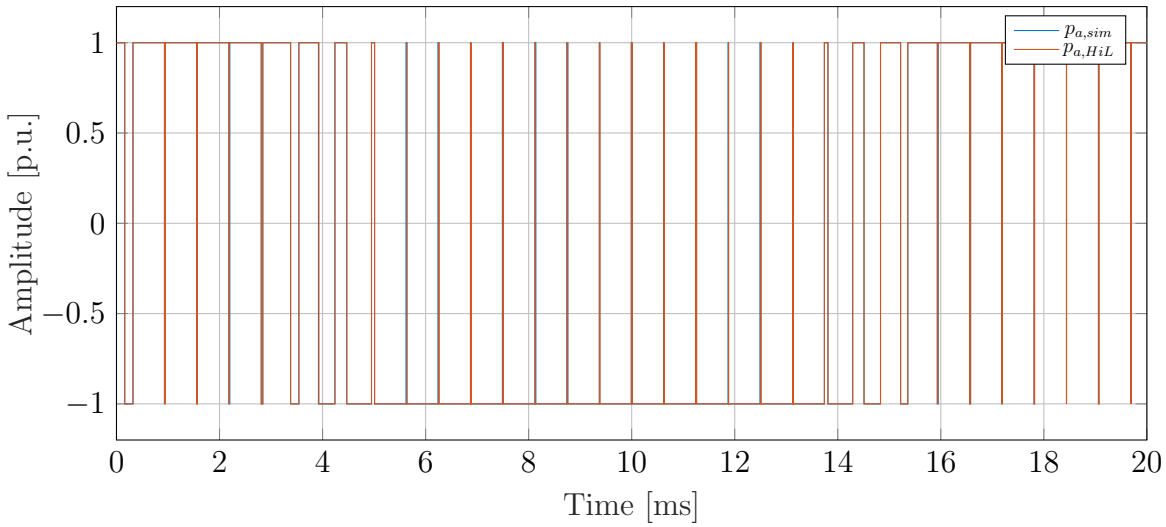


Figure 7.6: Comparison of PWM pulses generated by FPGA controller and MATLAB simulation for the proposed controller.

For the HiL simulation, the state variables consisted of 18 bit signals, the same accuracy as the controller. In order to see a significantly more accurate result, the simulation signals would require an additional 14 fractional bits. As is, the controller consists of 76 DSP slices and the HiL simulation uses up another 66, totalling 142 DSP slices. To test whether the accuracy of the HiL simulation could be improved, the fractional part of the signals were increased by only 6 bits each. The resulting synthesis failed because the amount of DSP slices required was higher than the 220 available on the device. A possible solution is to split the calculations between a couple of clock cycles to decrease the resources required. The more expensive solution is to use a device with more resources.

The second contributor to the slight offset in PWM pulses is the values calculated by the delay compensator process. Since the delay compensator calculates the state one controller period in advance, and the simulation matrices are discretised at a much smaller interval, the resulting values at the controller sampling instants do not match up exactly. The following example illustrates this principle. Let's say that the controller state model is discretised at T_c and the simulation state model is discretised at T_s. By using the same starting value for $\boldsymbol{x}(k)$, (4.5) is applied once to calculate the value of $\boldsymbol{x}(k+1)$ at the next controller sampling instant. The simulation runs 50 times in between controller sampling instants. The fixed point error accumulates for every step of the simulated time step T_s to get to the next controller time step T_c.

The third contributor to the difference in PWM pulses is the accuracy of the carrier signal. The carrier is generated by initializing a signal to -1 and then incrementing the value at each clock cycle by the appropriate value. When all the values add up, the quantisation error also causes a slight difference in amplitude when compared to the MATLAB equivalent. One possible solution to this problem is to store the values of the carrier (with high accuracy) over one fundamental cycle in a lookup table on the FPGA and then loading the values at every clock cycle.

The fourth contributor to the difference come from the fact that MATLAB simulation runs

sequentially, i.e. the controller calculates the optimal reference signals which is followed directly by the state update. The simulation "stands still" while the controller calculates the optimal PWM reference signals. In contrast, the HiL on the FPGA runs in parallel with the controller, which also affects the accuracy of the signals.

7.4 Implementation of HiL for LQR controller

7.4.1 Overview

The HiL implementation of the LQR controller follows the same principles as the proposed controller. The full system of the LQR controller including the HiL process is depicted in Fig 7.7. Once again, each box depicts a specific process and each arrow indicates a signal connecting two processes.

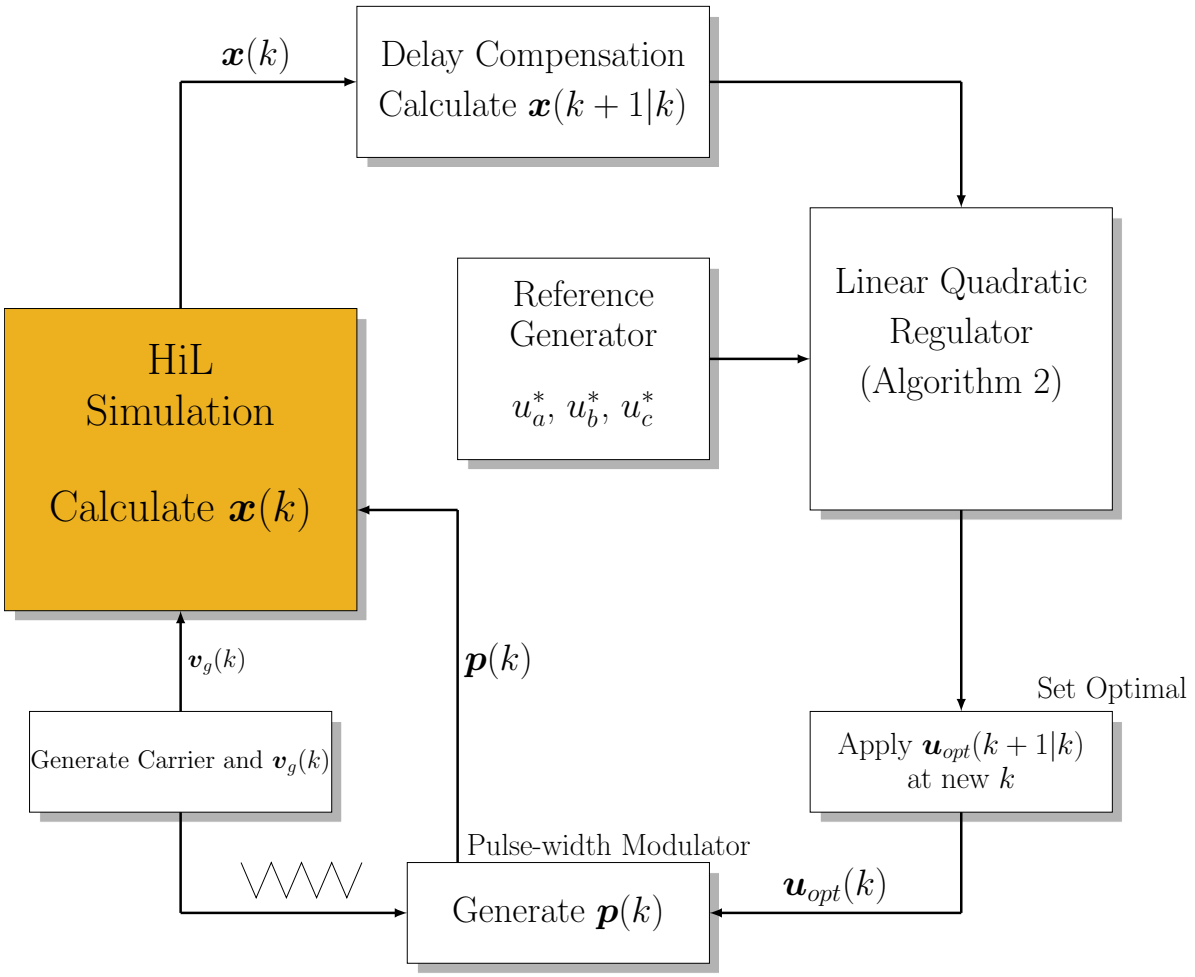


Figure 7.7: Flow diagram of LQR controller and HiL simulation.

The system is briefly explained below:

- A reference generator process generates the appropriate reference signals $\boldsymbol{u}_{abc}^*$ by loading the pre-calculated values from lookup tables in the FPGA.
- The delay compensator calculates one sample $\boldsymbol{x}(k+1|k)$ ahead, based on the current value of the HiL simulation.
- The LQR controller receives $\boldsymbol{x}(k+1|k)$ and $\boldsymbol{u}_{abc}^*$, and applies Algorithm 2 in Section 6.3 which calculates the adjusted optimal reference signals $\boldsymbol{u}_{opt}(k+1|k)$. These values are stored until the next controller sampling instant.

- At the next controller sampling instant, the values $\boldsymbol{u}_{opt}(k+1|k)$ are applied to the pulse-width modulator, which also receives the carrier signal. The pulse-width modulator generates PWM pulses $\boldsymbol{p}(k)$.

- The HiL simulation process receives the PWM pulses $\boldsymbol{p}(k)$ and the grid voltage signals $\boldsymbol{v}_g(k)$ and simulates the performance of the converter with LCL-filter.

7.4.2 HiL Simulation Results for LQR Controller

For the HiL implementation of the LQR controller, the same approach was taken as with the proposed controller. The HiL results are once again compared to the MATLAB equivalent, and is shown below. Fig 7.8 shows the converter currents, Fig 7.9 shows the grid currents and Fig 7.10 shows the filter capacitor voltages. The same notation in the legends are used as in the previous section.

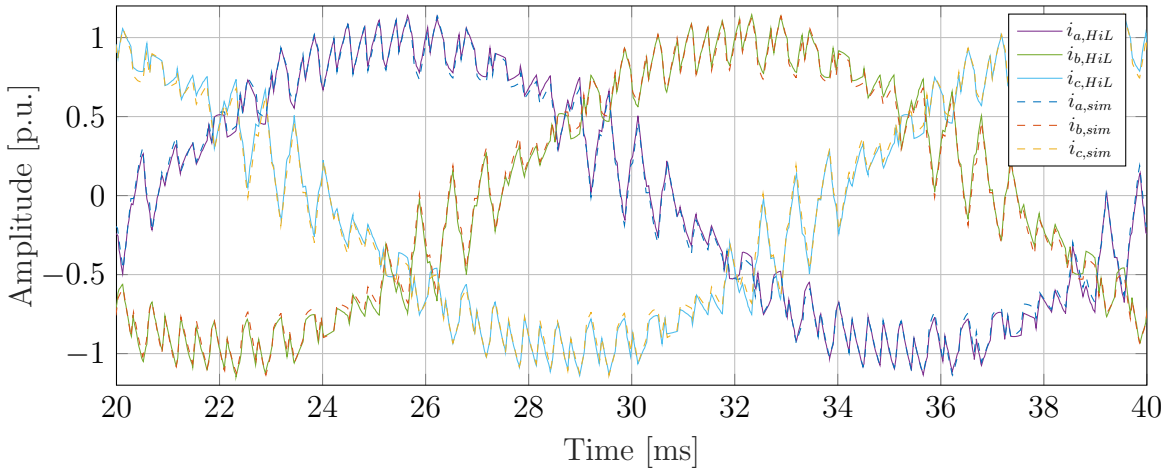


Figure 7.8: Comparison of converter current signals from HiL simulation and from MATLAB based simulation for the LQR controller.

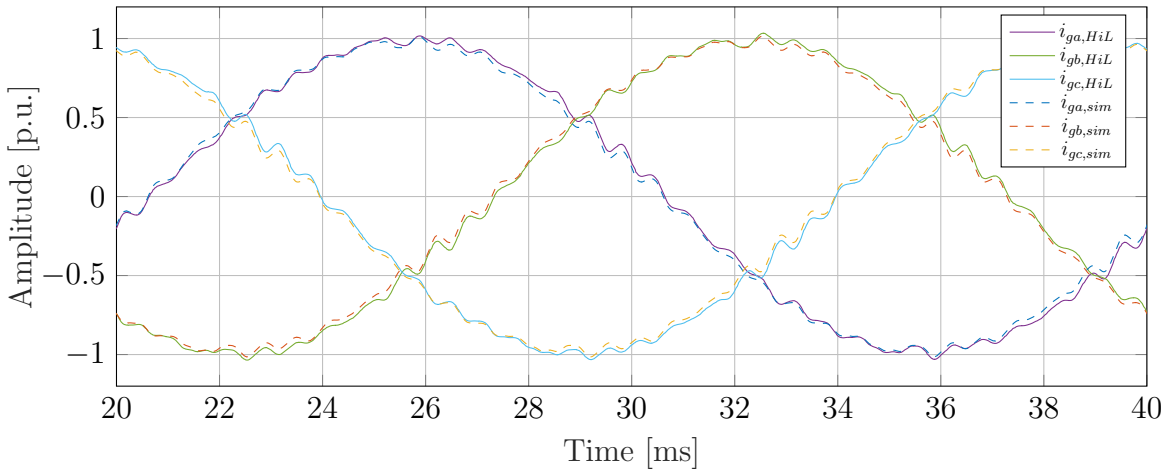


Figure 7.9: Comparison of grid current signals from HiL simulation and from MATLAB based simulation for the LQR controller.

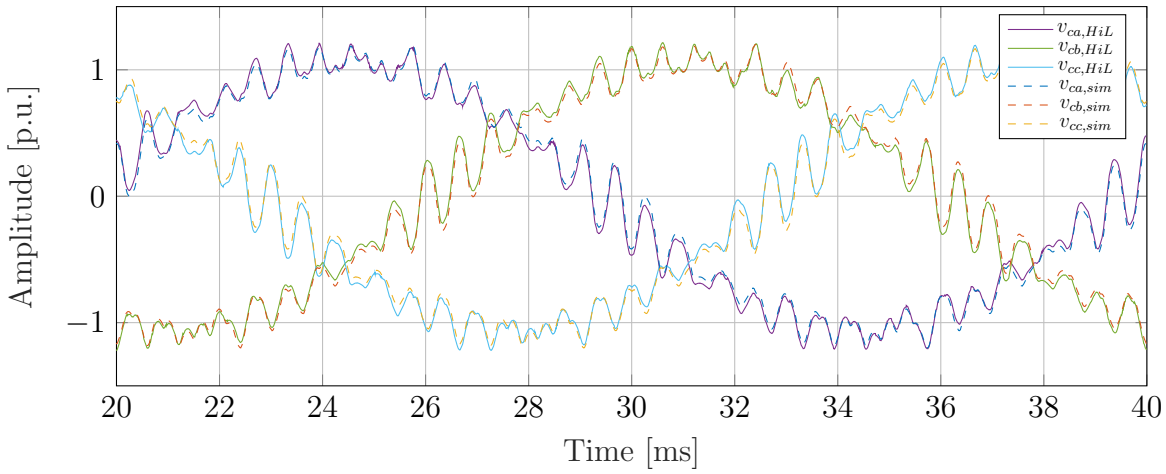


Figure 7.10: Comparison of capacitor voltage signals from HiL simulation and from MATLAB based simulation for the LQR controller.

The resulting signals are as expected using the same number of fractional bits as with the proposed controller implementation. Slight variations in PWM pulses create an offset between the MATLAB simulated results and the HiL signals. The same issues arise with this HiL simulation as mentioned previously with the proposed controller HiL simulation.

7.5 Summary

This chapter starts by describing the general approach of a HiL simulation, where the *simulation tracker* is discussed that keeps track of the HiL simulation process. The following section gives an overview of the specific implementation of a HiL simulation for the proposed controller. The results of a FPGA based HiL simulation is then compared to the equivalent MATLAB simulations. The results compared well, but for a slight offset in the PWM pulses generated by the controller.

The second part of this chapter discusses the implementation of the LQR controller on the FPGA device. An overview is given, after which the results is once again compared to the MATLAB equivalent. The results compare well with a slight offset caused by the difference in PWM pulses.

It was concluded that the discrepancy between the HiL and MATLAB based results was due to a combination of the quantisation error, the delay compensator process, the accuracy of the carrier signal, and the parallelism of the HiL simulation.

Chapter 8

Conclusion and Recommendations

8.1 Overview of Findings

8.1.1 Simulation Results

In Chapter 5, the two different control strategies were tested by means of a MATLAB-based simulation. The performance of both controllers are summarised below:
The first part focused on the proposed controller, which proved to be quite effective, with a settling time of $t = 2.5\,\text{ms}$ after a step in grid current reference was applied. The grid current has very low THD of 0.66% at steady-state nominal grid current. When the controller regulates the grid current at 0.5pu, the THD rises to only 1.09%. The robustness of the controller is also tested where the grid inductance of the system model is halved. This causes the THD to rise to 1.43%, due to the drop in attenuation of the current harmonics. The simulations also confirm that the controller is able to operate at a frequency less than twice the resonant frequency of the *LCL*-filter, whereas PI-based controllers require a switching frequency of at least three times the resonance.

Different weight factors for λ_u and lengths of the prediction horizon N_p were tested and plotted against the resulting THD of the grid current. It was found that much longer horizons for this particular controller do not reduce the harmonic content significantly. A shorter horizon of $N_p = 5$ was then chosen for the FPGA implementation, as it had a slightly lower THD than $N_p = 14$ for a specific value of λ_u. The lower prediction horizon also decreases the dimension of the control problem, in turn decreasing the computational burden.

The second part investigated the behaviour of the small-signal LQR controller. Since the controller is already developed and simulated in [8], only the main results are provided in this thesis. A step in grid current reference is applied at time $t = 25\,\text{ms}$. The controller settles quickly within $t = 3\,\text{ms}$. The LQR controller boasts the same advantage of the proposed controller, namely that it can operate at a switching frequency of less than twice that of the resonant frequency of the *LCL*-filter. One drawback is that the LQR controller cannot include constraints.

Both controller simulation sets of system parameters were tuned to achieve the lowest harmonic distortion. This makes it difficult to compare the two approaches and, therefore, a better comparison will have to be done in future research.

8.1.2 FPGA Implementation

Chapter 6 discussed the implementation of the two controllers in VHDL. Throughout the chapter, the focus is kept on the resource and time usage of the designs. Strategies were also discussed on how to effectively implement a controller with VHDL. Some key focuses were the use of simplified equations, the use of adder trees, fixed point versus floating point operations and delay compensation. The proposed controller was designed to use only 76 of the 220 DSP slices available on the Xilinx XC7Z020-1CLG484C Zynq-7000 FPGA. The LQR controller required much less DSP slices since the only multiplications needed were that of the feedback gain matrix $\boldsymbol{K}_f$.

8.1.3 HiL Implementation and Results

Chapter 7 consists of two parts. The first part discussed the implementation of a HiL simulation for the proposed controller in VHDL and the second discusses the same process for the LQR controller.

In the first part, a system overview was given, where all the processes involved in the design of the proposed controller HiL simulation were explained. The controller was given stationary test input signals to test the functionality. The resulting output of the controller was then plotted agains the MATLAB equivalent and it was found that the controller performs well on the FPGA, with no timing constraints violated and using only a small amount of the DSP resources (35% of the total available). The HiL was then connected to the controller and the resulting signals were plotted against the equivalent MATLAB generated simulation results. The controller acted as expected, with results close to that of the MATLAB signals.

The second part of the chapter provides an overview of the HiL design for the LQR controller. The resulting signals from the HiL simulation are plotted against the MATLAB equivalent. Once again, the signals of the HiL came close to the simulation, save for slight discrepancies.

The discrepancies between the HiL results and the MATLAB generated signals, in both designs, were attributed to a combination of factors that affected the PWM pulses. These factors include the quantisation error invoked when using a fixed point representation. The delay compensation process also adds a slight offset when calculating the states one full control sample ahead in time. The accuracy of the carrier signal generated on the FPGA also affected the PWM pulses.

8.2 Recommendations for Future Work

8.2.1 Inclusion of Grid Harmonics in the Model

The effects of grid harmonics on the converter currents should be considered, as it creates unwanted harmonic content in the controller. One possible approach is to implement a proportional-integral (PI) controller to suppress specific harmonics. Another strategy, as mentioned earlier and used in [3], involves the implementation of a virtual resistance (VR) in series with the grid inductor in order to attenuate the grid harmonics.

8.2.2 Optimisation of HiL Simulation

As mentioned in Section 7.3.2, the HiL simulation signals did not fit the MATLAB equivalent signals exactly. One possible way of improving the HiL, is by splitting the state update calculations over a few clock cycles. This will allow for the increase in the number of fractional bits allocated for the state variables, thus increasing the accuracy. This solution might, however, require an additional delay compensator, to compensate for the calculation delay.

8.2.3 Inclusion of Selective Harmonic Suppression

In order to comply with emission regulations, a future model of the proposed controller could include FFT calculations in the predictions. Specific harmonics can then be constrained. A feasibility study for such a controller could be done on the ZedBoard.

8.2.4 Practical Implementation

The HiL results prove that both controllers are able to function in real-time on an FPGA device. The next step is to obtain practical results to confirm the controller's efficacy in practise.

Bibliography

[1] South African Bureau of Standards (on behalf of NRS Project), *Electricity supply: quality of supply*. Pretoria: Standards South Africa, 2003.

[2] H. Miranda, R. Teodorescu, P. Rodriguez, and L. Helle, "Model predictive current control for high-power grid-connected converters with output *LCL* filter," in *2009 35th Annual Conference of IEEE Industrial Electronics*, 2009, pp. 633–638.

[3] J. Scoltock, T. Geyer, and U. K. Madawala, "A model predictive direct current control strategy with predictive references for MV grid-connected converters with *LCL*-filters," *IEEE Transactions on Power Electronics*, vol. 30, no. 10, pp. 5926–5937, 2015.

[4] N. Panten, N. Hoffmann, and F. W. Fuchs, "Finite control set model predictive current control for grid-connected voltage-source converters with *LCL* filters: A study based on different state feedbacks," *IEEE Transactions on Power Electronics*, vol. 31, no. 7, pp. 5189–5200, 2016.

[5] D. K. Yoo, L. Wang, E. Rogers, and W. Paszke, "Model predictive control of three phase voltage source converters with an *LCL* filter," in *2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE)*. IEEE, 2014, pp. 562–567.

[6] S. Mariéthoz and M. Morari, "Explicit model-predictive control of a PWM inverter with an *LCL* filter," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 2, pp. 389–399, 2009.

[7] S. Vazquez, C. Montero, C. Bordons, and L. G. Franquelo, "Model predictive control of a VSI with long prediction horizon," *International Symposium on Industrial Electronics (ISIE)*, pp. 1805–1810, 2011.

[8] T. Mouton and T. Geyer, "Trajectory-based LQR control of a grid-connected converter with an *LCL* filter," *Proc. of the IFAC Conference on Nonlinear Model Predictive Control, Madison, WI, USA, Aug*, 2018.

[9] T. Geyer, *Model Predictive Control of High Power Converters and Industrial Drives*. John Wiley & Sons, Nov. 2016.

[10] E. Twining and D. G. Holmes, "Grid current regulation of a three-phase voltage source inverter with an LCL input filter," in *2002 IEEE 33rd Annual IEEE Power Electronics Specialists Conference. Proceedings (Cat. No.02CH37289)*, vol. 3, Jun. 2002, pp. 1189–1194 vol.3.

[11] Poh Chiang Loh and D. G. Holmes, "Analysis of multiloop control strategies for LC/CL/LCL-filtered voltage-source and current-source inverters," *IEEE Transactions on Industry Applications*, vol. 41, no. 2, pp. 644–654, Mar. 2005.

[12] P. A. Dahono, "A control method to damp oscillation in the input LC filter," in *2002 IEEE 33rd Annual IEEE Power Electronics Specialists Conference. Proceedings (Cat. No.02CH37289)*, vol. 4, Jun. 2002, pp. 1630–1635 vol.4.

[13] J. Dannehl, M. Liserre, and F. W. Fuchs, "Filter-Based Active Damping of Voltage Source Converters WithLCLFilter," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 8, pp. 3623–3633, Aug. 2011.

[14] M. Liserre, A. Dell'Aquila, and F. Blaabjerg, "Genetic algorithm-based design of the active damping for an LCL-filter three-phase active rectifier," *IEEE Transactions on Power Electronics*, vol. 19, no. 1, pp. 76–86, Jan. 2004.

[15] "IEEE Recommended Practice and Requirements for Harmonic Control in Electric Power Systems," *IEEE Std 519-2014 (Revision of IEEE Std 519-1992)*, pp. 1–29, Jun. 2014.

[16] "Electromagnetic compatibility (EMC) - Part 2-4: Environment - Compatibility levels in industrial plants for low-frequency conducted disturbances," *IEC 61000-2-4*, Sep. 2002.

[17] D. G. Holmes and T. A. Lipo, *Pulse Width Modulation for Power Converters: Principles and Practice*. John Wiley & Sons, Oct. 2003.

[18] H. W. v. d. Broeck, H. Skudelny, and G. V. Stanke, "Analysis and realization of a pulsewidth modulator based on voltage space vectors," *IEEE Transactions on Industry Applications*, vol. 24, no. 1, pp. 142–150, Jan. 1988.

[19] J. Holtz, "Pulsewidth modulation-a survey," *IEEE Transactions on Industrial Electronics*, vol. 39, no. 5, pp. 410–420, Oct. 1992.

[20] D. G. Holmes, "The general relationship between regular-sampled pulse-width-modulation and space vector modulation for hard switched converters," in *Conference Record of the 1992 IEEE Industry Applications Society Annual Meeting*, Oct. 1992, pp. 1002–1009 vol.1.

[21] M. Dorfling, H. Mouton, P. Karamanakos, and T. Geyer, "Experimental evaluation of sphere decoding for long-horizon direct model predictive control," in *2017 19th European Conference on Power Electronics and Applications (EPE'17 ECCE Europe)*, Sep. 2017.

[22] B. Hassibi and H. Vikalo, "On the sphere-decoding algorithm I. Expected complexity," *IEEE Transactions on Signal Processing*, vol. 53, no. 8, pp. 2806–2818, Aug. 2005.

[23] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, Mar. 2004.

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

[25] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[26] J. M. Geldenhuys, H. du Toit Mouton, A. Rix, and T. Geyer, "Model predictive current control of a grid connected converter with *LCL*-filter," in *2016 IEEE 17th Workshop on Control and Modeling for Power Electronics (COMPEL)*, 2016, pp. 1–6.

[27] G. Goodwin, M. M. Seron, and J. A. de Dona, *Constrained Control and Estimation: An Optimisation Approach*. Springer, Mar. 2006.

[28] S. Richter, "Computational complexity certification of gradient methods for real-time model predictive control," Doctoral Thesis, ETH Zurich, 2012.

[29] D. G. Holmes, "The significance of zero space vector placement for carrier-based PWM schemes," *IEEE Transactions on Industry Applications*, vol. 32, no. 5, pp. 1122–1129, 1996.

[30] Xilinx, "Zynq-7000 SoC Data Sheet: Overview (DS190)," Jul. 2018.

[31] R. Jasinski, *Effective coding with VHDL: principles and best practice*. MIT Press, 2016.

[32] Avnet, "ZedBoard Hardware User's Guide," vol. 2.2, Jan. 2014.

[33] T. Geyer and D. E. Quevedo, "Multistep finite control set model predictive control for power electronics," *IEEE Transactions on Power Electronics*, vol. 29, no. 12, pp. 6836–6846, 2014.

[34] J. M. C. Geldenhuys, "Model Predictive Control of a Grid-connected Converter with *LCL*-filter," Master's Thesis, Stellenbosch University, 2018.

Appendices

Appendix A

Alternative optimisation approach

A second variation of the gradient projection method involves a transformation of $\boldsymbol{u}$ from abc to the $\alpha\beta$ plane. The constraints change from a cube in three dimensions to a hexagon in a two-dimensional space. The hexagonal shape can also be formed by inspection of all the possible space vectors formed by the switching states (see [29]). These space vectors with magnitude $M = \frac{4}{3}$ are shown in red in Fig. A.1, which also shows the feasible region of the solution as hexagon $\mathcal{P}_1$. The optimisation problem can be formulated as

$$\underset{\boldsymbol{U}(k)}{\text{minimise}} \;\; \tfrac{1}{2}\boldsymbol{U}^\top(k)\boldsymbol{H}\boldsymbol{U}(k) + \boldsymbol{\Theta}^\top(k)\boldsymbol{U}(k) \;\; \text{s.t. } \boldsymbol{\Omega}\boldsymbol{u}(k+i) \leq \boldsymbol{\rho} \;\; \forall \;\; i = 0, \ldots, N_p - 1,$$

where $\boldsymbol{\Omega}$ and $\boldsymbol{\rho}$ are defined as

$$\boldsymbol{\Omega} = \begin{bmatrix} -\sqrt{3} & -\frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} & 0 & 0 & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \sqrt{3} & \sqrt{3} & -1 & -1 & -\sqrt{3} \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & -1 \end{bmatrix}^\top \quad \text{and}$$

$$\boldsymbol{\rho} = \begin{bmatrix} -\sqrt{3} & \frac{M}{\sqrt{3}} & -\frac{M}{\sqrt{3}} & \frac{2}{\sqrt{3}} & -\frac{2}{\sqrt{3}} & \frac{M}{\sqrt{3}} & -\frac{M}{\sqrt{3}} & \sqrt{3}M & -\sqrt{3}M & -\frac{M}{2} & \frac{M}{2} & \sqrt{3}M \end{bmatrix}^\top.$$

Algorithm 5 receives $\boldsymbol{H}$, $\boldsymbol{\Theta}$, an initial guess $\boldsymbol{u}_0$ and the constant L. The algorithm is limited to a certain number of iterations N_f.

All polygons outside the hexagon $\mathcal{P}_1$, form the infeasible region and each requires a projection step towards the feasible region. For all $\mathcal{P}_i$ with even $i \in \{2, 4, \ldots, 12\}$, the iterate should be projected back to the nearest vertex. Otherwise, when an iterate falls within $\mathcal{P}_i$ with odd $i \in \{3, 5, \ldots, 13\}$, it should be projected back perpendicular onto the corresponding edge of the hexagon. A representation of the two types of projections is shown in Fig. A.1. The algorithm, therefore, needs to perform a search in order to determine the appropriate active polyhedron.

Fig. A.2 depicts the full set of line constraints that form the polygons in Fig. A.1. For each polygon, a unique sequence of 1s, -1s, and 0s can be defined by using their position with respect to a line. The value 1 is assigned when a polygon's position is above[1] a line. The value -1 is assigned when a polygon is below a line, and a 0 is assigned as a

[1]The terminology 'above' and 'below' indicates orientation with respect to a line, e.g. indicated as a $+$ and $-$ for lines 3, 7, and 10 in Fig. A.2.

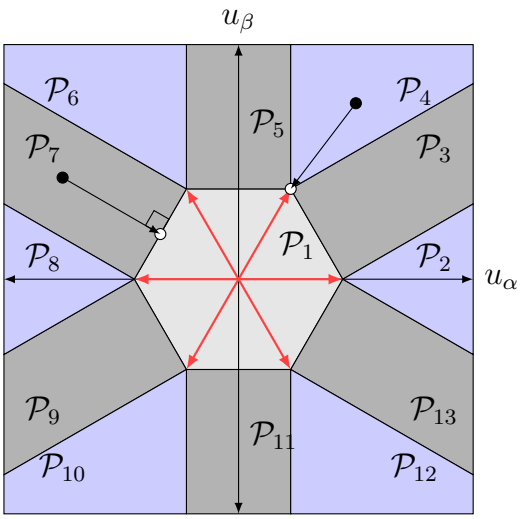


Figure A.1: Representation of projection principle for hexagonal constraints.

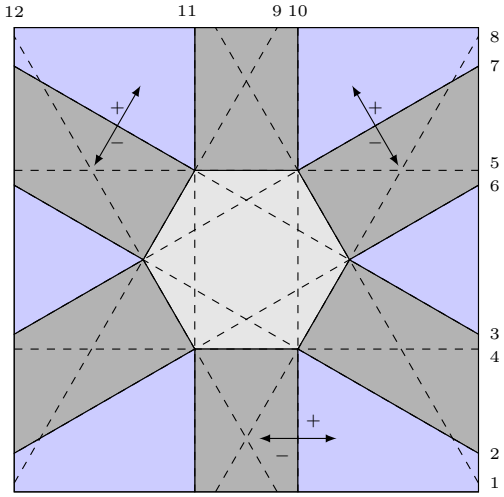


Figure A.2: Polygon search lines definition.

don't care when a line intersects the polygon. The sequence for polygon $\mathcal{P}_9$, e.g. can be written as $S_{\mathcal{P}_9} = [-1 \;\; -1 \;\; -1 \;\; 0 \;\; -1 \; 1 \;\; -1 \; 1 \;\; 0 \;\; -1 \;\; -1 \;\; -1]^\top$. Once all unique sequences are known, a simple logic test $\boldsymbol{\Omega u}(k) \leq \boldsymbol{\rho}$ yields a logic vector $\boldsymbol{\Xi}$ of 1s and 0s for every $\boldsymbol{u}$ over N_p, where all 0 values are replaced with -1s. The new logic vector is then multiplied element-wise (which is known as the Hadamard product and denoted with $\circ$) with each polygon sequence until all the entries of the resulting vector are non-negative. This indicates the active polyhedron.

For example, for $\boldsymbol{u}_1 = [-M \;\; -\frac{M}{2}]^\top$ (within $\mathcal{P}_9$), $\boldsymbol{\Xi}_{\boldsymbol{u}_1} \circ S_{\mathcal{P}_{10}} = [0\,0\,0\,{-1}\,0\,{-1}\,0\,0\,0\,0\,0\,0]^\top$, indicates an incorrect polyhedron since some entries are negative. This is the case for all polyhedra, except for $\mathcal{P}_9$, for which $\boldsymbol{\Xi}_{\boldsymbol{u}_1} \circ S_{\mathcal{P}_9}$ yields a vector with all entries non-negative.

Note that the matrices $\boldsymbol{\Omega}$ and $\boldsymbol{\rho}$ form redundant hyperplanes, but are required to describe the 12 unique infeasible polyhedra for the proposed algorithm. Since the constraints on each $\boldsymbol{u}$ in $\boldsymbol{U}$ are independent from one another, the problem can be broken down into N_p simple two-dimensional projections, instead of a single $2N_p$-dimensional projection.

Algorithm 5 Gradient Projection Algorithm

function GRADIENT PROJECTION METHOD($\boldsymbol{H}, \boldsymbol{\Theta}, \boldsymbol{u}_0, L$)
  for $i = 1 : N_f$ **do** ▷ For a predetermined number of iterations.
    $\boldsymbol{U}_i \leftarrow \boldsymbol{U}_{i-1} - \frac{1}{L}(\boldsymbol{H}\boldsymbol{U}_{i-1} + \boldsymbol{\Theta})$ ▷ Step in steepest descent.
    for $n = 1 : N_p$ **do** ▷ For each $\boldsymbol{u}$ in $\boldsymbol{U}$.
      $\boldsymbol{\Xi} \leftarrow \boldsymbol{\Omega}\boldsymbol{u}(n) \leq \boldsymbol{\rho}$ ▷ Generate logic vector.
      for $j = 1 : 13$ **do** ▷ Search through 13 unique sequences.
        if $(\boldsymbol{\Xi} \circ S_{\mathcal{P}_j}) \geq \mathbf{0}$ **then**
          $\mathcal{P}_{act} \leftarrow \mathcal{P}_j$ ▷ Active polygon found.
          Break
        end if
      end for
      if $\mathcal{P}_{act} = \mathcal{P}_i$, with $i \in \{2, 4, \ldots, 12\}$, **then**
        $\boldsymbol{u}(n) \leftarrow$ Nearest vertex
      else if $\mathcal{P}_{act} = \mathcal{P}_i$, with $i \in \{3, 5, \ldots, 13\}$, **then**
        $\boldsymbol{u}(n) \leftarrow$ Perpendicular to closest edge
      end if
    end for
  end for
end function

Appendix B

Derivation of Mathematical System Model

The chapter provides the detailed derivation of the mathematical system model of the grid connected two-level voltage source inverter with an LCL filter. For convenience, the system model diagram is again given below.

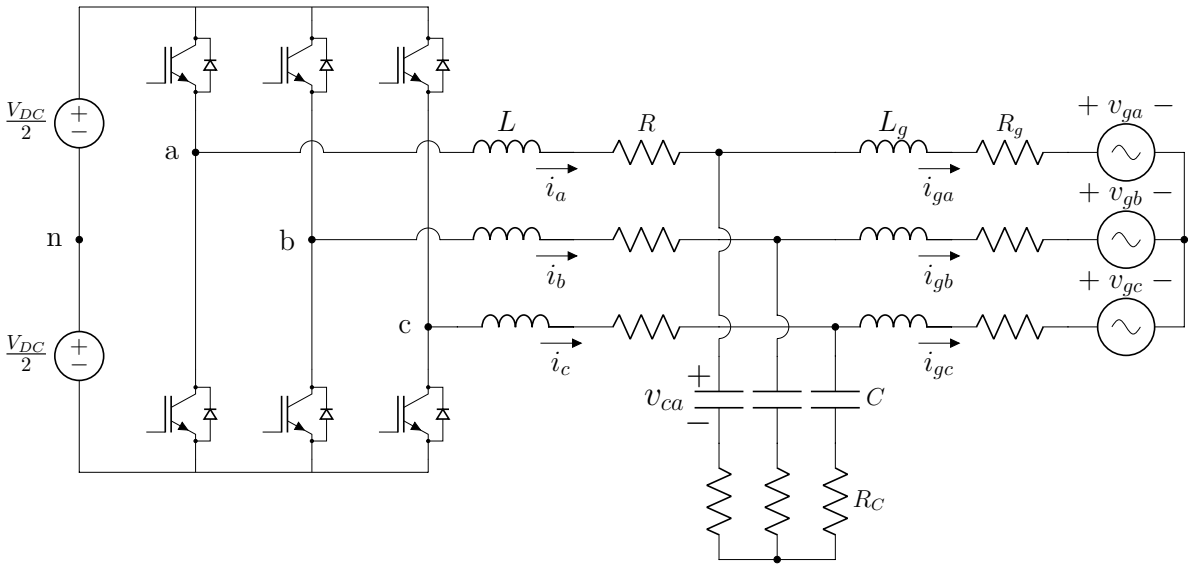


Figure B.1: Topology of grid-connected inverter via an LCL-filter

The inverter currents $\boldsymbol{i}_{abc}$, the grid currents $\boldsymbol{i}_{g,abc}$ and capacitor voltages $\boldsymbol{v}_{c,abc}$ are chosen as the state variables. By examining the per phase equivalent of the circuit in Fig. B.1, and by applying Kirchoff's voltage and current laws (KVL and KCL), the mathematical system model can be described by the following differential equations

$$
\begin{aligned}
v_{g,a} - R_c C\frac{dv_{c,a}}{dt} + v_{c,a} + i_{g,a}R_g + L_g\frac{di_{g,a}}{dt} &= 0 \\
v_{c,a} + R_c C\frac{dv_{c,a}}{dt} - \frac{V_{DC}}{2}p_a + L\frac{di_a}{dt} + i_a R &= 0 \\
-i_a + C\frac{dv_{c,a}}{dt} + i_{g,a} &= 0
\end{aligned}
$$

These equations can be rewritten with the state variables on the left-hand side as

$$\begin{aligned}
\frac{di_a}{dt} &= -\frac{i_a(R+R_g)}{L} + \frac{i_{g,a}R_c}{L} - \frac{v_{c,a}}{L} + \frac{V_{DC}p_a}{2L} \\
\frac{di_{g,a}}{dt} &= \frac{i_aR_c}{L_g} - \frac{i_{g,a}(R_c+R_g)}{L_g} + \frac{v_{c,a}}{L_g} - \frac{v_{g,a}}{L_g} \\
\frac{dv_{c,a}}{dt} &= \frac{i_a}{C} - \frac{i_{g,a}}{C}
\end{aligned}$$

The mathematical system model can be reduced by transforming the three-phase quantities into the stationary $\alpha\beta$ reference frame. The state vector is defined as

$$\boldsymbol{x} = \begin{bmatrix} i_\alpha & i_\beta & i_{g\alpha} & i_{g\beta} & v_{c\alpha} & v_{c\beta} \end{bmatrix}^\top .$$

The grid voltage vector $\boldsymbol{v}_{g,abc}(t)$ and switching state vector $\boldsymbol{p}(t)$ are included as inputs to the system. The continuous-time state-space equations describing the system can be expressed as

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{F}\boldsymbol{x}(t) + \boldsymbol{G}\boldsymbol{p}(t) + \boldsymbol{P}\boldsymbol{v}_{g,abc}(t) ,$$

where

$$\boldsymbol{F} = \begin{bmatrix}
\frac{R+R_C}{-L} & 0 & \frac{R_C}{L} & 0 & \frac{1}{-L} & 0 \\
0 & \frac{R+R_C}{-L} & 0 & \frac{R_C}{L} & 0 & \frac{1}{-L} \\
\frac{R_C}{L_g} & 0 & \frac{R_g+R_C}{-L_g} & 0 & \frac{1}{L_g} & 0 \\
0 & \frac{R_C}{L_g} & 0 & \frac{R_g+R_C}{-L_g} & 0 & \frac{1}{L_g} \\
\frac{1}{C} & 0 & \frac{1}{-C} & 0 & 0 & 0 \\
0 & \frac{1}{C} & 0 & \frac{1}{-C} & 0 & 0
\end{bmatrix} ,$$

$$\boldsymbol{G} = \begin{bmatrix}
\frac{V_d}{2L} & 0 \\
0 & \frac{V_d}{2L} \\
0 & 0 \\
0 & 0 \\
0 & 0 \\
0 & 0
\end{bmatrix} \text{ and } \boldsymbol{P} = \begin{bmatrix}
0 & 0 \\
0 & 0 \\
\frac{1}{-L_g} & 0 \\
0 & \frac{1}{-L_g} \\
0 & 0 \\
0 & 0
\end{bmatrix} \boldsymbol{K} .$$

Appendix C

Cost Function Derivation in Vector Form

The cost function derivation follows the procedure in [9] with the addition of the grid voltage to the model.

The tracking error between reference signals and state variables as well as a weight λ_u on the change in reference signal $\boldsymbol{u}(k)$ are included as control objectives [33]. By using the state-space equation (3.21), the model-predictive controller predicts a trajectory over a predefined prediction horizon N_p and yields the sequence of reference vectors $\boldsymbol{u}(k)$ that minimises the cost function. The cost function over the prediction horizon is defined as

$$\begin{aligned} J &= \sum_{\ell=k}^{k+N_p-1} ||\boldsymbol{x}^*(\ell+1) - \boldsymbol{x}(\ell+1)||^2_{\boldsymbol{Q}} + \lambda_u ||\boldsymbol{\Delta u}(\ell)||_2^2 \\ &= J_1 + J_2 \ , \end{aligned}$$

where $\boldsymbol{Q}$ denotes the weight matrix. The variable $\boldsymbol{x}^*$ indicates the state reference vector and the difference of two successive PWM reference vectors $\boldsymbol{\Delta u}(\ell)$ is defined as

$$\boldsymbol{\Delta u}(\ell) = \boldsymbol{u}(\ell) - \boldsymbol{u}(\ell-1) \ ,$$

The term $\boldsymbol{u}(k)$ denotes a single PWM reference vector. The first term J_1 can be expanded by applying (3.21) to the state vector consecutively over the prediction horizon [26], yielding

$$\boldsymbol{x}(k+m) = \boldsymbol{A}^m \boldsymbol{x}(k) + \sum_{\ell=0}^{m-1} \boldsymbol{A}^{m-1-\ell} \boldsymbol{B u}(k+\ell) + \sum_{\ell=0}^{m-1} \boldsymbol{A}^{m-1-\ell} \boldsymbol{\mathcal{V}} \boldsymbol{v}_{g,abc}(k+\ell)$$

with $m = \ 1, \cdots, N_p - 1$. The output trajectory is defined as

$$\boldsymbol{X}(k) = [\boldsymbol{x}^\top(k+1) \ \boldsymbol{x}^\top(k+2) \cdots \boldsymbol{x}^\top(k+N_p)]^\top$$

and the corresponding reference trajectory is denoted as $\boldsymbol{X}^*(k)$. The output trajectory $\boldsymbol{X}(k)$ can be written in matrix format as

$$\boldsymbol{X}(k) = \boldsymbol{\Gamma u}(k) + \boldsymbol{\Upsilon U}(k) + \boldsymbol{\Omega V}_g(k) \ , \tag{C.1}$$

where $\mathbf{\Gamma}$, $\mathbf{\Upsilon}$ and $\mathbf{\Psi}$ are defined in Section 4.3. The resulting first term is now

$$J_1 = ||\mathbf{\Gamma}\boldsymbol{x}(k) + \mathbf{\Upsilon}\boldsymbol{U}(k) + \mathbf{\Omega}\boldsymbol{V}_g(k) - \boldsymbol{X}^*(k)||^2_{\boldsymbol{Q}_c} \ ,$$

where $\boldsymbol{Q}_c = \text{diag}(\boldsymbol{Q},\ \boldsymbol{Q}, \cdots \boldsymbol{Q})$ is an expansion of the weight matrix over the prediction horizon.
The second term J_2 can be written in vector format by the following expansion

$$\begin{aligned} J_2 &= \sum_{\ell=k}^{k+N_p-1} \lambda_u ||\boldsymbol{\Delta u}(\ell)||_2^2 \\ &= \lambda_u \sum_{\ell=k}^{k+N_p-1} (u(\ell) - u(\ell-1))^\top (u(\ell) - u(\ell-1)) \\ &= \lambda_u (\boldsymbol{SU}(k) - \boldsymbol{Eu}(k-1))^\top (\boldsymbol{SU}(k) - \boldsymbol{Eu}(k-1)) \\ &= \lambda_u ||\boldsymbol{SU}(k) - \boldsymbol{Eu}(k-1)||_2^2 \ , \end{aligned}$$

where $\boldsymbol{S}$ and $\boldsymbol{E}$ are defined in Section 4.3. The sequence of PWM reference vectors over the prediction horizon is defined as

$$\boldsymbol{U}(k) = [\boldsymbol{u}(k)^\top\ \boldsymbol{u}(k+1)^\top \cdots \boldsymbol{u}(k+N_p-1)^\top]^\top \ ,$$

where $\boldsymbol{U}(k)$ is a $[2N_p \times 1]$ vector. The resulting cost function can now be written as

$$J = ||\mathbf{\Gamma}\boldsymbol{x}(k) + \mathbf{\Upsilon}\boldsymbol{U}(k) + \mathbf{\Omega}\boldsymbol{V}_g(k) - \boldsymbol{X}^*(k)||^2_{\boldsymbol{Q}_c} + \lambda_u ||\boldsymbol{SU}(k) - \boldsymbol{Eu}(k-1)||_2^2 \ . \qquad \text{(C.2)}$$

The cost function in (C.2) can be written in a quadratic program by using some algebraic manipulation. The derivation can be found in [34], where the only difference is the inclusion of the weight matrix $\boldsymbol{Q}_c$ in this thesis.

Appendix D

Resources used in Project

The development board that was used during the course of this project is the ZedBoard, and is shown in Fig. D.1.

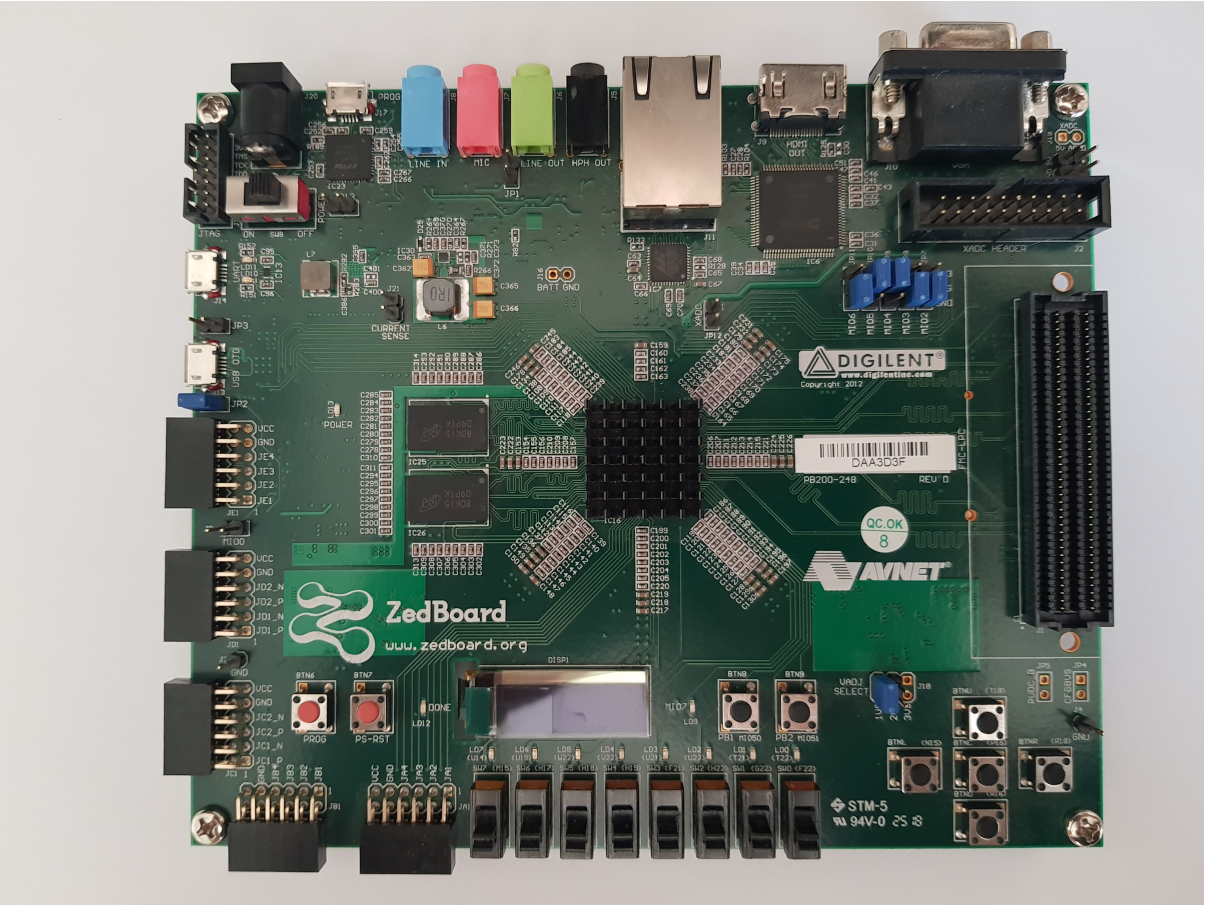


Figure D.1: Photo of the ZedBoard used for prototyping.

The project summary of the Vivado IDE is shown in Fig. D.2. The summary supplies useful information, such as the worst negative slack (WNS) and the utilisation of resources. The negative slack indicates the time at which a certain task must be executed to prevent the following tasks from being delayed.

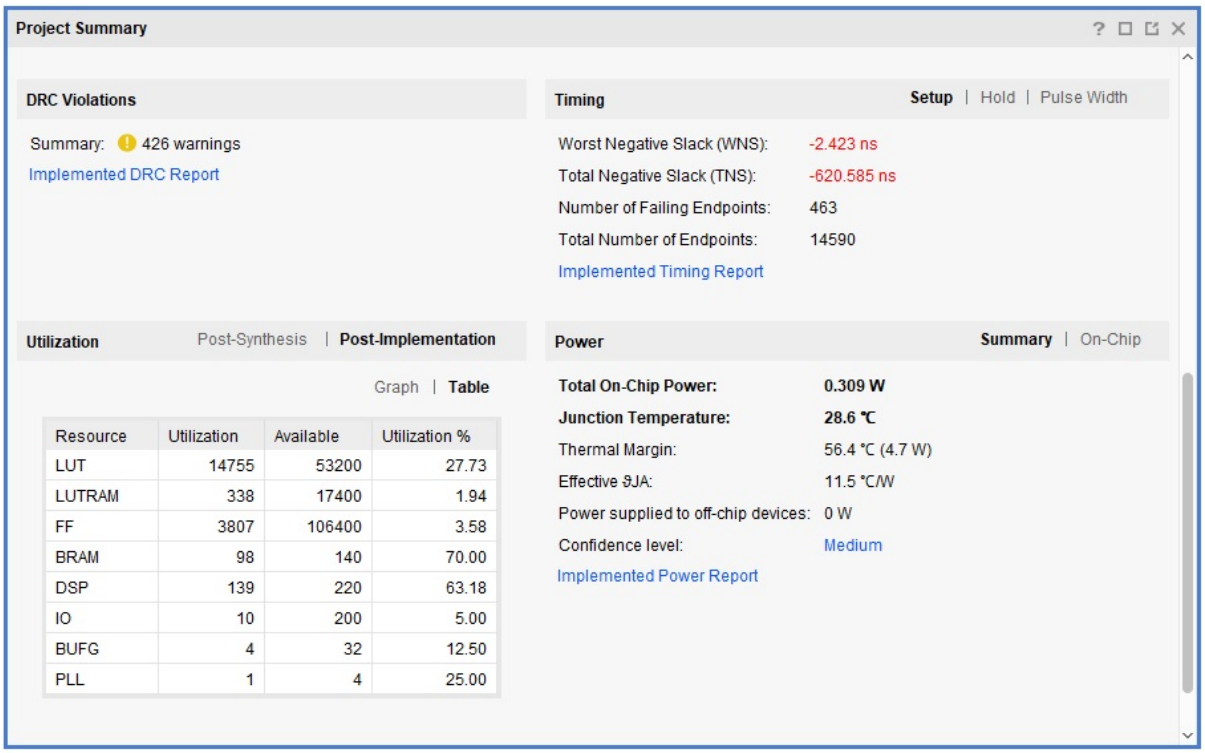


Figure D.2: Screenshot of project summary in Vivado IDE.