

# Alternative Sphere Decoding for Finite Control Set Model Predictive Control of Power Electronic Converters

by

Johannes Hendrikus Raath



Dissertation presented in fulfilment of the requirements for the degree of  
Doctor of Philosophy in Electrical Engineering in the Faculty of  
Engineering at Stellenbosch University

Supervisor: Prof Hendrik du Toit Mouton

Co-supervisor: Prof Tobias Geyer

March 2021

# Declaration

By submitting this report 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.

Date: .....  
March 2021

Copyright © 2021 Stellenbosch University  
All rights reserved.

# Uittreksel

## Alternatiewe Sfeer Dekodering vir Eindige Beheerstel Modelvoorspellendebeheer van Drywingselektroniese Omskakelaars

J.H. Raath

*Departement Elektriese en Elektroniese Ingenieurswese,  
Universiteit van Stellenbosch,  
Privaatsak X1, Matieland 7602, Suid Afrika.*

Proefskrif: PhD (Ing)

Maart 2020

Die dinamiese vooruitgang in rekenaartegnologie die afgelope dekade het modelvoorspellendebeheer (MVB) na vore laat tree as 'n belowende alternatiewe beheertegniek vir drywingselektronika-toepassings. Die neiging in die drywingselektronikaveld is eindige beheerstel modelvoorspellendebeheer, waardeur 'n enkele veelvoudige-inset veelvoudige-uitset beheerder tot stand gebring kan word om ingewikkelde beheerlusse te vervang en beheerseine vir direkte toepassing aan die omskakelaar te genereer. Langhorison-MVB verwys na die evaluering van die optimaliseringskriterium oor 'n voorspellingshorison van meer as een. Dit is veral gesog vir die geslote lusprestasie wat dit tydens bestendige toestande bied. Langhorison MVB benodig egter meer berekeninge om die optimale beheeraksie te vind. Die berekeningslas geassosieer met MVB kan wel verminder word deur wiskundige programmering, waar die onderliggende MVB-probleem geformuleer word as 'n heeltallige-minimumkwadraat probleem. Met betrekking tot drywingselektroniese stelsels, het 'n onlangse aanpassing van die sfeer-dekoderingsalgoritme na vore gekom as 'n baie gewilde keuse vir die oplossing van die heeltallige-minimumkwadraat probleem.

Die primêre bydrae van hierdie tesis is om 'n alternatiewe dekoderingsstrategie te ontwerp wat die heeltallige-minimumkwadraat probleem met betrekking tot eindige beheerstel MVB vir meervlakomskakelaars sal oplos. Eksponensiële ruimte-algoritmes word ondersoek en gekoppel aan 'n toepaslike drywingselektroniese toepassing. 'n Sekondêre bydrae van die tesis is 'n vooraf-kondisioneringsalgoritme of projeksie-algoritme wat voorgestel word. Eienskappe wat moderne prosesseringstechnologie bevoordeel, word vasgelê en toegepas op die veelbekroonde bol-dekoderingsalgoritme om 'n dekoderingsbenadering voor te stel wat matriksberekeninge insluit. Dit maak voorsiening vir die doeltreffende voorafberekening en berging van stelselmatrikse vir latere aanlyngebruik. Die voorgestelde algoritmes word toegepas in 'n voorspellende beheerder vir 'n induksiemotordryf, en word in werklike tyd geëvalueer.



# Abstract

## Alternative Sphere Decoding for Finite Control Set Model Predictive Control of Power Electronic Converters

J.H. Raath

*Department of Electrical and Electronic Engineering,  
University of Stellenbosch,  
Private Bag X1, Matieland 7602, South Africa.*

Dissertation: PhD (Eng)

March 2020

The rapid advance in computational power over the past decade has enabled model predictive control (MPC) to emerge as a promising alternative control technique for fast reacting power electronic applications. Trending in the power electronic system field is finite control set MPC, whereby a single multiple-input multiple-output (MIMO) controller can be realised to replace complicated PI control loops and deliver control signals for direct application to the converter. As an extension of basic MPC, long-horizon MPC refers to the evaluation of the optimisation criterion over a prediction horizon of more than one. Long-horizon MPC is desirable for its enhanced closed-loop performance during steady-state operation. Unfortunately, long-horizon MPC requires more computations to find the optimum control action, especially in FCS-MPC, as the computational burden increases exponentially with an extension of the prediction horizon. The computational burden associated with long-horizon MPC can be reduced through mathematical programming where the underlying MPC problem is re-formulated as an integer least-squares problem. With regard to power electronic systems in particular, a recent adaptation of the Sphere

Decoding Algorithm (SDA) has emerged as a very popular choice for solving or decoding of the ILS problem.

The main objective of this thesis is to devise an alternative decoding strategy that will solve the ILS problem relating to long-horizon FCS-MPC for multi-level inverters. Exponential space algorithms are investigated and matched to an appropriate power electronic application. From the investigation, a novel preconditioning algorithm or projection algorithm evolved. Characteristics that favour modern processing technology are captured and applied to the well-acclaimed SDA to propose a decoding approach that involves matrix calculations. This also permits the efficient precomputation and storage of system matrices offline for online use. The proposed algorithms are incorporated into a predictive controller for a power electronic drive and are evaluated in real time.

# Acknowledgements

First and foremost, I would like to thank my Supervisor, Prof Mouton for his courage in taking on an “older” student. Knowing that it would not be an easy task, he patiently guided and mentored me throughout the years. Secondly, I enjoyed the privilege of having Prof Tobias Geyer as my Co-supervisor. His insight and wealth of knowledge kept me humble but excited to explore and learn more.

Secondly, I would like to thank my employer, Central University of Technology, Free State, and in particular Prof Herman Vermaak, who always helped when challenges emerged. I also want to thank my colleagues, Dr Ben Kotze, for introducing me to Prof Mouton, and Dr Michelle Erasmus, who assisted with my mathematical woes.

Lastly, I want to thank the most important people in my life, who sacrificed, supported and motivated me throughout this study: My two amazing sons, “Drix en Pina - julle is ysters!”. To my wonderful parents, Hennie (R.I.P.) and Joey, “Dankie vir die krag in my, Pa” en “Ma, dankie vir die absolute voorbeeld van liefde”.

# Contents

<b>Declaration</b>	<b>i</b>
<b>Uittreksel</b>	<b>ii</b>
<b>Abstract</b>	<b>iv</b>
<b>Acknowledgements</b>	<b>vi</b>
<b>Contents</b>	<b>vii</b>
<b>List of Figures</b>	<b>xii</b>
<b>List of Tables</b>	<b>xviii</b>
<b>Nomenclature</b>	<b>xx</b>
Notation . . . . .	xx
List of symbols . . . . .	xxi
List of acronyms . . . . .	xxiii
<b>1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	1
1.1.1 Control strategies in power electronics . . . . .	2
1.1.2 MPC as control method . . . . .	4
1.2 Objectives . . . . .	7
1.3 Thesis outline . . . . .	8
1.3.1 Chapter 1: Introduction . . . . .	8
1.3.2 Chapter 2: Preliminaries . . . . .	8
1.3.3 Chapter 3: Search space and solvers . . . . .	8

1.3.4	Chapter 4: Performance estimation and development . . . . .	9
1.3.5	Chapter 5: Alternative sphere decoder . . . . .	9
1.3.6	Chapter 6: Real-time verification . . . . .	9
1.3.7	Chapter 7: Summary . . . . .	10
<b>2</b>	<b>Preliminaries</b>	<b>11</b>
2.1	The control problem . . . . .	11
2.2	Model predictive control . . . . .	12
2.2.1	Fundamentals of MPC . . . . .	12
2.3	The optimisation problem underlying FCS-MPC . . . . .	15
2.3.1	Formulation of the cost function . . . . .	15
2.3.2	Optimisation problem . . . . .	19
2.4	Physical systems . . . . .	20
2.4.1	Three phase systems . . . . .	20
2.5	Mathematical sub-system models . . . . .	22
2.5.1	Multilevel inverters . . . . .	22
2.5.2	Linear circuit elements . . . . .	27
2.5.3	Induction machine . . . . .	28
2.5.4	Per unit definitions . . . . .	29
2.5.5	Induction machine model . . . . .	30
2.5.6	Predictive control of an induction machine . . . . .	32
2.5.7	Drive performance parameters . . . . .	33
2.5.8	Chapter summary . . . . .	34
<b>3</b>	<b>Search space and solvers</b>	<b>35</b>
3.1	Lattice fundamentals . . . . .	35
3.2	Lattice structure . . . . .	38
3.3	Closest vector problem . . . . .	42
3.4	Space partitioning . . . . .	43
3.5	Decoding strategies . . . . .	46
3.5.1	Sphere decoding algorithm . . . . .	46
3.5.2	Iterative slicing algorithm . . . . .	52
3.5.3	Micciancio Voulgaris algorithm . . . . .	58

3.6	Conditioning of the CVP target . . . . .	62
3.6.1	Proposed projection algorithm . . . . .	64
3.6.2	Concept . . . . .	65
3.6.3	Projection algorithm . . . . .	68
3.6.4	Optimality . . . . .	70
3.6.5	Projection algorithm with enlargement factor . . . . .	73
3.7	Summary . . . . .	75
<b>4</b>	<b>Performance estimation and development</b>	<b>76</b>
4.1	Inverter with $RL$ -load . . . . .	76
4.1.1	System model . . . . .	77
4.1.2	MPC formulation . . . . .	79
4.2	Simulation: performance evaluation . . . . .	79
4.2.1	Steady-state conditions . . . . .	81
4.2.2	Transient conditions . . . . .	84
4.3	Summary . . . . .	88
<b>5</b>	<b>Alternative sphere decoder</b>	<b>90</b>
5.1	Sphere block decoding . . . . .	91
5.1.1	Concept . . . . .	91
5.2	Algorithm . . . . .	95
5.2.1	Complexity . . . . .	97
5.3	Performance simulations . . . . .	100
5.4	Multilevel considerations . . . . .	102
5.5	Summary . . . . .	104
<b>6</b>	<b>Real-time verification</b>	<b>106</b>
6.1	Introduction . . . . .	106
6.2	Case study: Induction machine drive . . . . .	107
6.3	Control system . . . . .	107
6.4	Plant dynamics . . . . .	109
6.5	FCS-MPC . . . . .	111
6.5.1	Controller formulation . . . . .	111
6.5.2	Unconstrained minimum . . . . .	112

6.5.3	Practical considerations of the MPC process . . . . .	113
6.5.4	Controller settings . . . . .	116
6.6	Experimental setup . . . . .	121
6.6.1	Real-time simulation system . . . . .	122
6.6.2	Drive system realisation . . . . .	124
6.7	Steady-state performance . . . . .	126
6.8	Transient performance . . . . .	128
6.8.1	Load torque step . . . . .	128
6.8.2	Speed reference step . . . . .	130
6.8.3	Computational benchmark . . . . .	130
6.9	Increased voltage levels . . . . .	132
6.9.1	Experimental results . . . . .	132
6.9.2	Computational performance . . . . .	134
6.10	Summary . . . . .	135
<b>7</b>	<b>Conclusions</b>	<b>136</b>
7.1	Contributions . . . . .	136
7.1.1	Chapter 3 . . . . .	136
7.1.2	Chapter 4 . . . . .	137
7.1.3	Chapter 5 . . . . .	137
7.1.4	Chapter 6 . . . . .	138
7.2	Publications . . . . .	139
7.3	Future work . . . . .	139
	<b>Appendices</b>	<b>141</b>
<b>A</b>	<b>Linear algebra</b>	<b>142</b>
A.1	Projection onto a vector . . . . .	142
A.2	Hyperplanes . . . . .	144
A.3	Projection onto a plane . . . . .	145
A.4	Transforming normals . . . . .	146
A.5	Hypercube elements . . . . .	147
A.6	Dual interpretation of projection . . . . .	149
A.7	Orthogonal decomposition . . . . .	149

<i>CONTENTS</i>	<b>xi</b>
<b>B Algorithms and processes</b>	<b>150</b>
B.1 Adapted projection algorithm . . . . .	150
B.2 Acquiring the output reference variables . . . . .	151
<b>C Real-time simulation model</b>	<b>154</b>
C.0.1 Console sub-system . . . . .	156
C.0.2 Master sub-system . . . . .	157
C.0.3 Slave sub-system . . . . .	163
<b>Bibliography</b>	<b>166</b>



# List of Figures

1.1	Power electronic converter control strategies. . . . .	3
2.1	Typical control structure of a power electronics system. . . . .	11
2.2	Typical MPC structure in the field of power electronics. . . . .	13
2.3	Neutral-point-clamped inverter topology. . . . .	23
2.4	Cascaded H-bridge inverter topology. . . . .	24
	(a) A single H-bridge cell. . . . .	24
	(b) A series of H-bridge cells. . . . .	24
2.5	Voltage space vectors generated by a three-level NPC inverter in the $\alpha\beta$ plane along with their congruent three-phase switch positions ("+" refers to "1" and "-" to "-1"). . . . .	26
2.6	Linear circuits. . . . .	27
	(a) $RL$ load. . . . .	27
	(b) $LC$ low-pass filter. . . . .	27
2.7	Equivalent circuit of an induction machine in the $dq$ reference frame. . . . .	29
2.8	Basic Predictive Current Control scheme. . . . .	33
3.1	Square lattice $\mathbb{Z}^2$ with two different sets of basis vectors. . . . .	36
	(a) Orthogonal basis. . . . .	36
	(b) Skewed basis. . . . .	36
3.2	<i>Truncated lattice</i> with orthonormal basis $\mathbf{I}_d$ in $\mathbb{R}^2$ . . . . .	38
3.3	A truncated lattice with skew basis in $\mathbb{R}^2$ . Bounding the convex hull is a set of $(2 \times d)$ , $(d - 1)$ -dimensional affine hyperplanes. . . . .	39
3.4	Lattice span as intersection of half-spaces in $\mathbb{R}^2$ . . . . .	42
3.5	Voronoi diagram for the set of $HU$ -sites in $\mathbb{R}^2$ space. . . . .	44
3.6	Voronoi diagram of a lattice with basic Voronoi cell and relevant vectors. . . . .	45

3.7	Lattice points $\mathbf{H}[1\ 1]^T$ and $\mathbf{H}[1\ 0]^T$ enclosed by a sphere with radius $\delta$ from the target $\mathbf{x}$ . . . . .	47
3.8	Search tree for a two-dimensional ( $d = 2$ ) CVP with limited search space. . . . .	48
3.9	Slicing of the $\mathbf{H}$ -coordinate space with Voronoi border pairs located at $\pm\frac{1}{2}\mathbf{v}$ . . . . .	53
3.10	The error vector $\mathbf{e}_r$ as a result of the target $\mathbf{x}$ and a starting lattice point ( $\mathbf{s} = \underline{0}$ ). . . . .	54
3.11	Projection of $\mathbf{e}_r$ onto $\mathbf{v}_3$ resulting in a projection coefficient with a magnitude larger than 1.5 . . . . .	54
3.12	Updating the lattice point gives $\mathbf{s} = \underline{0} + 2\mathbf{v}_3$ , with the new error vector residing in slice $\pm\frac{1}{2}\mathbf{v}_3$ . Projection of $\mathbf{e}_r$ onto $\mathbf{v}_2$ results in a projection coefficient with a magnitude larger than 0.5, but less than 1.5. . . . .	55
3.13	Updating the lattice point gives $\mathbf{s} = \underline{0} + 2\mathbf{v}_3 - 1\mathbf{v}_2$ with the effected error vector residing in slices $\pm\frac{1}{2}\mathbf{v}_3$ and $\pm\frac{1}{2}\mathbf{v}_2$ . Projection of $\mathbf{e}_r$ onto $\mathbf{v}_1$ results in a projection coefficient with a magnitude less than 0.5, signifying containment by the slice $\pm\frac{1}{2}\mathbf{v}_1$ . . . . .	56
3.14	Orthogonal projection of $\mathbf{e}_r$ onto the set of Voronoi-relevant vectors to identify the optimum vector $\mathbf{v}^* = \mathbf{v}_2$ . . . . .	59
3.15	Updating the lattice point $\mathbf{s}$ with the optimum Voronoi-relevant vector $-\mathbf{v}_1$ , resulting in the error vector $\mathbf{e}_r \in \mathcal{V}$ . . . . .	60
3.16	Truncated lattice with target residing outside translations of the basic Voronoi cell and proposed orthogonal projection. . . . .	62
3.17	Effect of orthogonal projection onto the convex hull of the lattice. . . . .	63
3.18	Minimum distance projection vector onto a convex hull in $d$ -dimensional space. . . . .	65
3.19	Orthogonal decomposition of the projection vector in a $d$ -dimensional space. . . . .	66
3.20	Orthogonal projection onto the $(d - 1)$ -dimensional parallelotope in the subspace spanned by $\mathcal{H}_{1,0}$ . . . . .	67
3.21	Voronoi diagram with $\mathcal{V}(\mathcal{L}, \underline{0}) \not\subseteq \mathcal{P}$ . . . . .	71
3.22	Projection onto enlarged convex hull $P^*$ . . . . .	72
3.23	Backtracking on the projection trajectory, simulating an enlarged projection hull. . . . .	73
4.1	Three-phase 3L NPC VSC with $RL$ -load. . . . .	77
4.2	State machine representation of the MPC scheme with target preconditioning. . . . .	80

4.3	System waveforms for steady-state operation of the 3-level NPC inverter during steady-state conditions. (a) The inverter output voltage $\mathbf{v}_i$ and (b) the $RL$ -load current $\mathbf{i}_i$ . . . . .	81
4.4	Maximum number of iterations required by the respective algorithms during steady-state conditions. The sphere decoding algorithm with a preconditioned target is denoted by SDA*. . . . .	82
4.5	Maximum flops incurred during steady-state conditions. The sphere decoding algorithm with and without target preconditioning are respectively denoted by SDA* and SDA. Reference to the performance of the MVA or SDA* <i>combined with</i> the projection algorithm (PA), are respectively denoted by SDAp and MVAp. . . . .	83
	(a) Individual. . . . .	83
	(b) Cumulative. . . . .	83
4.6	Maximum algorithm termination times during steady-state conditions. The sphere decoding algorithm with and without target preconditioning is respectively denoted by SDA* and SDA. Reference to the performance of the MVA or SDA* <i>combined with</i> the projection algorithm (PA), is respectively denoted by SDAp and MVAp. . . . .	83
	(a) Individual. . . . .	83
	(b) Cumulative. . . . .	83
4.7	During transient conditions, (a) the inverter output voltage $\mathbf{v}_{abc}$ and (b) the $RL$ -load current $\mathbf{i}_{abc}$ . . . . .	84
4.8	Maximum number of iterations required by the respective algorithms during transient conditions. . . . .	85
4.9	Maximum flops incurred during transient conditions. . . . .	85
	(a) Individual. . . . .	85
	(b) Cumulative. . . . .	85
4.10	Average algorithm termination times per sample during transient conditions. . . . .	86
	(a) Individual. . . . .	86
	(b) Cumulative. . . . .	86

4.11	Average termination times of the MATLAB <sup>®</sup> “quadprog”, Multi-parametric toolbox (MPT) “distance” and Projection algorithm (ProjH) solvers for the minimum distance projection problem in steady-state and transient conditions(*). Steady-state results are indicated with solid lines and transient conditions with dashed lines. . . . .	87
5.1	Tree traversal through a three-dimensional tree structure (left) and one-dimensional tree structure (right) to the 3 <sup>rd</sup> horizon / 9 <sup>th</sup> dimension. . . . .	92
5.2	Computational lower bounds for the SDA and SBDA when solving MPC problems of horizon $N$ . . . . .	99
5.3	Performance of the decoding algorithms in steady-state conditions. The sphere decoding algorithm without target preconditioning is denoted by SDA. Performance of the SDA, MVA and SBDA <i>combined with</i> the projection algorithm, are respectively denoted by SDAp, MVAp and SBDAP. . . . .	101
	(a) Maximum flops. . . . .	101
	(b) Maximum termination times. . . . .	101
5.4	Performance during transient conditions. The sphere decoding algorithm without target preconditioning is denoted by SDA. Performance of the SDA, MVA and SBDA <i>combined with</i> the projection algorithm, are respectively denoted by SDAp, MVAp and SBDAP. . . . .	101
	(a) Maximum flops. . . . .	101
	(b) Maximum termination times. . . . .	101
5.5	Lattice structure in two dimensions of the solution space for the FCS-MPC problem of a five-level diode-clamped multilevel inverter. . . . .	102
5.6	Maximum flop counts effected by the CVP solvers for a number of multilevel inverters with voltage levels ranging from three to eleven. MPC prediction horizons of $N = 3$ and 4 are shown. . . . .	103
5.7	Maximum termination times of the algorithms to solve the optimization problem underlying MPC for prediction horizons of $N = 3$ and 4. . . . .	104
6.1	Three-phase three-level diode-clamped multilevel inverter driving an induction machine via an intermediate $LC$ filter. . . . .	107
6.2	Vector quantities of the plant in the rotating $dq$ -frame. Aligned with the rotor flux vector and rotating with $\omega_s$ . . . . .	114

6.3	Illustration of the theoretical and real-time approach to MPC with prediction horizon $N = 1$ . . . . .	114
6.4	Effect of switching frequency on the stator current total dynamic distortion for selected prediction horizons $N$ . The system sampling frequency is 8kHz. . . . .	117
6.5	Effect of sampling frequency ( $F_s$ ) on the stator current total demand distortion ( $I_{s,TDD}$ ) for selected prediction horizons ( $N$ ). . . . .	119
6.6	Relationship between the prediction horizon period ( $T_N$ ) and the stator current total demand distortion ( $I_{s,TDD}$ ) for selected prediction horizons ( $N$ ). . . . .	119
6.7	Effect of sampling frequency ( $F_s$ ) on the computational complexity (maximum flops) for selected prediction horizons ( $N$ ). . . . .	120
6.8	Relationship between the prediction horizon period ( $T_N$ ) and the computational complexity (maximum flops) for selected prediction horizons ( $N$ ). . . . .	120
6.9	Orthogonality defect of the generator matrix $\mathbf{H}$ for selected prediction horizons at various sampling frequencies. . . . .	121
6.10	OPAL-RT technologies real-time simulation system. . . . .	122
6.11	OPAL-RT technologies, OP5600 platform. . . . .	123
6.12	OP5600 internal layout. . . . .	124
6.13	Experimental setup. . . . .	125
6.14	System waveforms for steady-state operation of the three-level inverter drive at nominal speed and torque. Specific to the $N = 8$ case with an average device switching frequency of 300Hz. Rotor speed $\omega_r$ , developed torque $T_e$ , inverter voltage $\mathbf{v}_i$ , inverter current $\mathbf{i}_i$ , capacitor voltage $\mathbf{v}_c$ , stator current $\mathbf{i}_s$ and controller computation time $t_c$ . . . . .	127
6.15	System waveforms corresponding to a variation in the mechanical load from 0pu to 0.8pu. Developed torque $T_e$ , rotor speed $\omega_r$ , inverter current $\mathbf{i}_i$ , capacitor voltage $\mathbf{v}_c$ , stator current $\mathbf{i}_s$ and controller computation time $t_c$ . . . . .	129
6.16	System waveforms corresponding to a speed reference step from 0.2pu to 1pu. Rotor speed $\omega_r$ , developed torque $T_e$ , inverter current $\mathbf{i}_i$ , capacitor voltage $\mathbf{v}_c$ , stator current $\mathbf{i}_s$ and controller computation time $t_c$ . . . . .	131
6.17	SDA computation time for solving the $N = 8$ MPC problem. . . . .	132

6.18	System waveforms for steady state operation of the five-level CHB inverter drive at nominal speed and torque. Specific to the $N = 3$ case with an average device switching frequency of 150Hz. Rotor speed $\omega_r$ , developed torque $T_e$ , inverter voltage $\mathbf{v}_i$ , inverter current $\mathbf{i}_i$ , capacitor voltage $\mathbf{v}_c$ , stator current $\mathbf{i}_s$ and controller computation time $t_c$ . . . . .	133
A.1	Projection onto a vector in $\mathbb{R}^2$ space. . . . .	142
A.2	A hyperplane in $\mathbb{R}^3$ space. . . . .	144
A.3	Projection of a vector onto a plane in $\mathbb{R}^3$ space. . . . .	145
A.4	Hypercubes in dimensions 0 to 3. . . . .	147
A.5	Hypercube elements . . . . .	148
A.6	Dual interpretation of projection of points $\mathbf{x}_{1,2}$ onto the convex set $\mathcal{P} \in \mathbb{R}^2$ . . . . .	149
A.7	Orthogonal decomposition of vector $\mathbf{s} \in \mathbb{R}^3$ with subspace $S \in \mathbb{R}^2$ . . . . .	149
C.1	Drive system model. . . . .	155
C.2	Console sub-system: User interface. . . . .	156
C.3	Master sub-system: PI and predictive controller. . . . .	157
C.4	Conversion and estimation of the system states. . . . .	158
C.5	Rotor flux estimation . . . . .	158
C.6	FCS-MPC process. . . . .	159
C.7	Look-up tables selected from discretized speed measurement. . . . .	160
C.8	Compute output references in $dq$ -rotating frame and convert to stationary Alpha-Beta frame. . . . .	160
C.9	Extrapolation of the system states. . . . .	161
C.10	Calculate the unconstrained minimum. . . . .	161
C.11	Optimising the ILS problem. . . . .	162
C.12	Initial sphere radius computed from the Babai estimate. . . . .	162
C.13	Slave sub-system with Plant, I/Os and Data capture subsystems. . . . .	163
C.14	Plant: NPC inverter driving the IM via an intermediate $LC$ filter. . . . .	164
C.15	NPC inverter gating control. . . . .	164
C.16	Digital-to-analogue and analogue-to-digital conversion via the external hard-wired loop. . . . .	165

# List of Tables

2.1	NPC inverter leg state, switch states and output voltage level. . . . .	23
2.2	Cascaded H-bridge inverter cell leg state, switch states and output voltage level.	24
2.3	Base values of a pu system in terms of $V_R$ , $I_R$ , and $\omega_{sR}$ . . . . .	30
4.1	Algorithm denotations for experimental results. . . . .	80
5.1	Influence of block-size selection for a 3-level inverter. . . . .	99
6.1	Parameters of the drive system. . . . .	108
6.2	Idealistic performance metrics of the IM drive for steady-state operation (simulated in MATLAB <sup>©</sup> ). . . . .	122
6.3	Performance metrics of the drive system during steady-state operation (OPAL real-time simulation). . . . .	126
6.4	Performance metrics of the drive system during steady-state operation (OPAL real-time simulation). . . . .	134
6.5	Performance metrics of the drive system during steady-state operation (MATLAB <sup>©</sup> simulation). . . . .	134

# List of Algorithms

1	Sphere Decoding algorithm . . . . .	49
2	Iterative Slicing algorithm . . . . .	57
3	<i>Micciancio Voulgaris</i> CVP algorithm . . . . .	60
4	<i>Projection</i> algorithm . . . . .	68
5	<i>Projection</i> algorithm . . . . .	74
6	Sphere Block Decoding algorithm - main routine . . . . .	96
7	Sphere Block Decoding algorithm - subroutine . . . . .	97
8	<i>Projection</i> algorithm . . . . .	150



# Nomenclature

## Notation

Throughout this thesis, the characterization of mathematical symbols and definitions are as follow:

- Scalars are italic letters:  $y, Y$ ;
- Vectors are bold lower case letters:  $\mathbf{y}$ ;
- Matrices are bold upper case letters:  $\mathbf{Y}$ ;
- Sets are bold upper case calligraphic letters:  $\mathcal{Y}$ ;

Elements of vectors and matrices, as well as columns of matrices, are identified by subscripts in the format:

- For a matrix  $\mathbf{Y}$ ,  $\mathbf{y}_n$  denotes the  $n^{\text{th}}$  column vector with  $y_{m,n}$  denoting the element in row  $m$ , column  $n$ .
- The  $m^{\text{th}}$  element of a vector  $\mathbf{y}$  is denoted by  $y_m$ .
- On occasion will *MATLAB*<sup>©</sup> like notation be used to denote subsets of a matrix or vector. For example  $\mathbf{Y}_{:,n:k}$  defines a matrix constituted by the columns  $n, n + 1 \dots, k$  of the matrix  $\mathbf{Y}$ . Similar will a sub-vector of  $\mathbf{y}$ , formed by the elements  $m, m + 1 \dots, k$  be represented by  $\mathbf{y}_{m:k}$ .

## List of symbols

### Mathematical definitions

$\mathbb{Z}, \mathbb{Q}, \mathbb{R}, \mathbb{N}$	Integers, rational, real, natural numbers
$\{\dots\}$	Set
$\in$	Belongs to
$\subset$	Subset of
$\mathbf{x} \in \mathbb{R}^m$	Real vector of dimension $m$
$\mathbf{0}$	Zero vector in $\mathbb{R}^m$
$\mathbf{X} \in \mathbb{R}^{m \times n}$	Real matrix of dimension $m$ and rank $n$
$\mathbf{0}_m$	$m \times m$ Zero matrix
$\mathbf{I}_m$	$m \times m$ Identity matrix
$=$	Equality
$<, >, \leq, \geq$	Inequalities
$\approx$	Approximate
$\rightarrow$	Mapping
$\infty$	Infinity

### Mathematical operators

$\min\{.\}$	Minimization of the function $\{.\}$
$\max\{.\}$	Maximization of the function $\{.\}$
$\mathbf{x}^T, \mathbf{X}^T$	Transpose of a vector or matrix
$\langle \mathbf{x} \cdot \mathbf{x} \rangle$	Inner product $\mathbf{x}^T \mathbf{x}$
$\times$	Cartesian product
$ \mathbf{x} $	Absolute value
$\ \mathbf{x}\ _1$	Taxicab norm ( $l_1$ norm)
$\ \mathbf{x}\ _2$ or $\ \mathbf{x}\ $	Euclidean norm ( $l_2$ norm)
$\ \mathbf{x}\ _\infty$	Infinity norm ( $l_\infty$ norm)
$\ \mathbf{x}\ _\Lambda^2$	Euclidean norm squared and weighted with the matrix $\Lambda$
$\text{span}\{\mathcal{L}\}$	Span of a lattice

$\lfloor x \rfloor$	Closest integer to $x$ , $\lfloor 1/2 \rfloor = 1$
$\#\mathbf{x}, \#\mathcal{X}$	Cardinality of a vector or set
$\otimes, \oplus, \ominus, \odot$	Item-wise operations
$x!$	Factorial of $x$

**Variables**

$f_s$	sampling frequency
$T_s$	sampling interval
$\mathbf{x}, \mathbf{u}, \mathbf{y}$	state-, input- and output vectors.
$\mathbf{F}, \mathbf{G}, \mathbf{C}$	state-, input- and output matrices in continuous-time domain
$\mathbf{A}, \mathbf{B}, \mathbf{C}$	state-, input- and output matrices in discrete-time domain
$d$	dimension of a vector, matrix or space
$t$	continuous-time
$k$	discrete integer steps
$\mathcal{U}$	inverter-leg control set ( $\mathcal{U} \in \mathbf{U}$ )
$\mathbf{U}$	inverter control set
$N$	prediction horizon
$\mathbf{X}$	sequence of predicted state-vectors
$\mathbf{U}$	sequence of control-vectors ( $\mathbf{U} \in \mathbb{U}$ )
$\mathbb{U}$	feasible set of control-vector sequences
$\mathbf{Y}$	sequence of predicted output-vectors
$J$	performance index or cost
$\Lambda$	weighing matrix

## List of acronyms

ADC	analogue-to-digital converter
CCS	continuous control set
CCS-MPC	continuous control set MPC
CVP	closest vector problem
CVPP	CVP with preprocessing
DAC	digital-to-analogue converter
DPC	direct power control
DSP	digital signal processing
DSC	direct self control
DTC	direct torque control
DTMC	direct mean torque control
ESA	exhaustive search algorithm
FCS	finite control set
FCS-MPC	finite control set MPC
FOC	field-oriented control
FPGA	field programmable gate array
GCT	gate-commutated thyristor
IGCT	integrated gate commutated thyristor
ILS	integer least-squares
IM	induction machine
ISA	iterative slicing algorithm

LC	inductive-capacitive
LHMPC	long-horizon model predictive control
LTI	linear time-invariant
MIMO	multiple-input multiple-output
MPC	model predictive control
MPCC	model predictive current control
MPSC	model predictive speed control
MPTC	model predictive torque control
MVA	Micciancio Voulgaris algorithm
NNP	nearest neighbor problem
NPC	neutral-point clamped
NP-hard	non-deterministic polynomial-time hard
OD	orthogonality defect
PA	projection algorithm
PC	predictive control
PCC	predictive current control
PE	power electronic
PES	power electronic system
PI	proportional integral
PTC	predictive torque control
PWA	piecewise affine
PWM	pulse-width modulation
pu	per unit
QP	quadratic programming

RL	resistive-inductive
RMF	rotating magnetic field
rms	root-mean-square
RSA	randomized sieve algorithm
RTS	real-time simulation
SBDA	sphere block decoding algorithm
SDA	sphere decoding algorithm
SISO	single-input single-output
SVP	shortest vector problem
TDD	total demand distortion
THD	total harmonic distortion

# Chapter 1

## Introduction

### 1.1 Background

“POWER converters and drive control technology have been in continuous development since the second half of the 20th century and have proven to be enabling technologies in practically every application area” [1]. Driven by the economy of scale, several industrial processes have increased their power-level needs which triggered the development of power semiconductors, innovative converter topologies and advanced control methods.

With the advent of the 1990s semiconductors have evolved to high-power insulated-gate bipolar transistors IGBTs and gate-commutated thyristors (GCTs). The innovation of these switching devices have since made it the technology of choice for all medium- and high-power applications, mainly due to their high power density, superior switching characteristics and ease of control.

In the application areas of medium voltage and high power, multilevel inverter topologies have raised much attention due to their ability to deliver high output voltages [2]. The three most common multilevel inverter topologies are, the diode clamped, the flying capacitor, and the cascaded H-bridge [3, 4]. Multilevel inverters, however, have some disadvantages. One in particular is the increased number of semiconductor switches, each demanding a linked gate drive circuit. This may cause the overall system to be more expensive, and adds to the *complexity of the control strategy*. Much research has been done on multilevel converter topologies, and numerous control methods have been presented

in the literature. According to [5], the pivotal requirements for power converter control schemes are as follows:

- **Lower harmonic distortions** stem from accurate reference tracking and result in the reduction or removal of harmonic filters.
- **Reduced switching losses**, as a consequence of low switching frequencies, equal increased inverter efficiency.
- **High controller bandwidth** ensures fast closed-loop control.
- **Accurate load power control** for keeping the power within the preset limits.
- **Robustness to variations** in parameter, measurement and observer values.
- **Computational cost**, the last but defining factor for successful implementation which enable the controller to deliver all the above-mentioned requirements in *real time*.

### 1.1.1 Control strategies in power electronics

Several control schemes have been proposed for the control of power electronic systems and drives (PES). Some of these are shown in Fig.1.1 as presented by [6]. Hysteresis and linear controls with pulse-width modulation (PWM) dominate the literature in general. However, the explosive evolution of digital signal processing (DSP) over the last decade has enabled the implementation of increasingly more complex control schemes [1]. Some of these new schemes for power converters include fuzzy logic, sliding mode control, and *predictive control*.

Predictive control (PC) covers an extensive class of controllers with recent application in power converters. In general, it can be described as a predictive algorithm which predicts the future trajectories or response of the system using a model and then, based on a predefined optimality criterion specific to the model, selects the most appropriate control action [1]. The four basic strategies employed by predictive control algorithms for power electronic system are illustrated in Fig. 1.1.



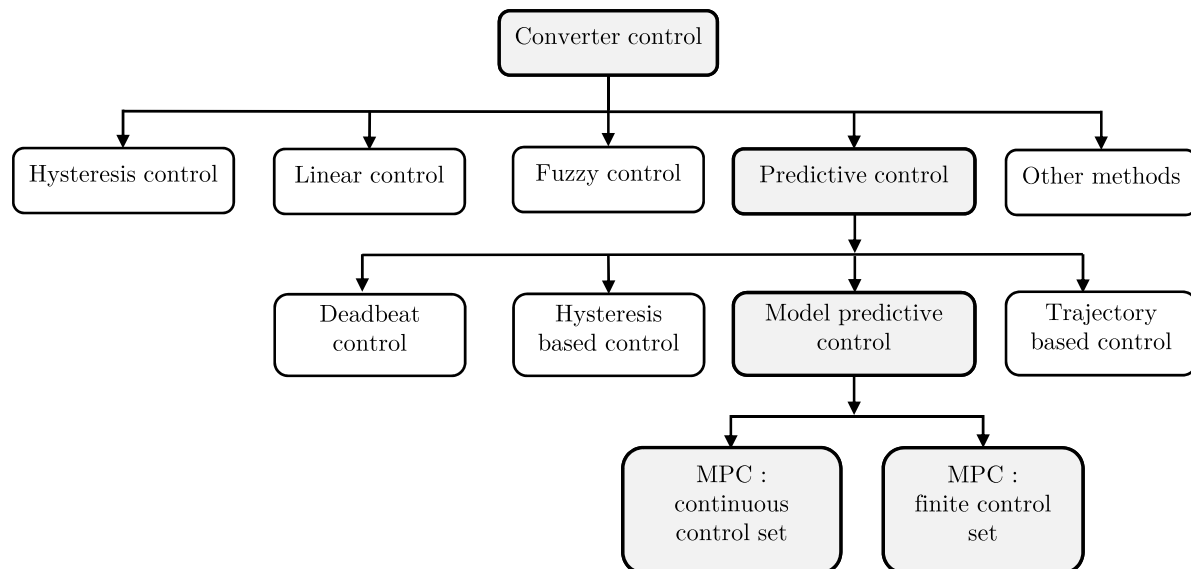


Figure 1.1: Power electronic converter control strategies.

**Deadbeat control** exhibits the property that for a given reference, the error between it and the output is minimised over a single sampling instant [7]. Importantly, for multivariable systems, deadbeat controllers may need more than one sampling instant to achieve the objective (reachability concept). Input constraints, also can limit this one sampling instant error minimization objective [8]. Deadbeat controllers have been used in a multiple of applications [9, 10, 11, 12] where a fast dynamic response is required.

**Hysteresis-based control** aims at maintaining the controlled system variables within the predefined boundaries of a hysteresis area or space. The bang-bang controller is a simple example of this approach. More eloquent applications include direct torque control (DTC) [13] and direct power control (DPC) [14] of an induction motor.

**Trajectory-based control** forces the system's controlled variables to follow pre-calculated trajectories. Direct self control (DSC) [15] is a well-known scheme that utilises this strategy. Another is direct mean torque control (DTMC) [16] which combines the hysteresis- and trajectory-based strategies.

**Model predictive control** employs a *flexible* optimisation criterion which is expressed as a performance index or cost function. Model predictive control (MPC) can be subdivided into MPC with a *continuous control set* (CCS-MPC), and MPC with a *finite control set* (FCS-MPC).

### 1.1.2 MPC as control method

The Model Predictive Control strategy has been discovered and reinvented several times over the past decades. MPC schemes developed in the late seventies have found wide acceptance in the process industry, mainly because of their complex models and slow speed dynamics [17]. Power electronic converters have faster dynamics compared to chemical processes; therefore, faster computers and improvements in computational efficiency were necessary to evolve and enable the use of MPC for power converters [18]. Some of the first MPC-related concepts for power converters were successfully implemented and experimentally verified in the 1980s [19].

As an alternative to conventional controllers, MPC offers several advantages compared to linear control schemes. Trending in the power electronic system field is finite control set MPC, whereby a single multiple-input multiple-output (MIMO) controller can be realised to resolve complicated proportional integral (PI) control loops and deliver control signals for direct application to the converter. The process is relatively straightforward, which enables the consideration of various non-linearities and constraints. According to [8], MPC offers several advantages that make it suitable for the control of power converters:

- **Concepts** are intuitive and comprehensible.
- **Multiple switches** and states can be managed.
- **Constraints** and non-linearities can be easily incorporated.
- **Controller** implementation is relatively easy.

As an extension of basic MPC, the concept of long-horizon MPC (LHMPC) refers to the evaluation of the optimisation criterion over a horizon, i.e. a sample period of more than one. Long-horizon MPC is desirable for its enhanced closed-loop performance during steady-state operation. Geyer (2011) [20], has shown that long prediction horizons could lead to a significant increase in performance during *steady-state* operating conditions, lowering the current distortions and the converter switching frequency. Another important characteristic of long-horizon MPC is its ability to counter system stability issues [21, 22, 23].

Unfortunately, long-horizon MPC requires more computations to obtain the optimum solution, which in a real-time application should occur within a sampling interval [24, 25]. When extending the prediction horizon, the computational burden increases exponentially, especially in FCS-MPC that contain integer values. An additional issue is the problem of low time resolution, as switch transitions can only be invoked at the sampling intervals. Mainly for these reasons have implementations of FCS-MPC typically been characterised by a prediction horizon that is almost always set to one [26]. Fundamental to the computational burden is the optimisation criterion and complexity of the system. In literature a couple of long-horizon MPC strategies can be found with a modest computational burden. These, include: move blocking used in a dc-dc converter [27], extrapolation [28] and event-based horizon [29], both applied in medium voltage ac drives. The main reason why so few practical MPC implementations are found in literature is due to the computational issue.

Standard MPC solves the optimisation problem *online*, and with the commonly used approach of *exhaustive enumeration* long-horizon MPC is impractical, as the optimal solution cannot be found in real time. The advent of more powerful processing platforms in the form of digital signal processors (DSPs) and field-programmable gate arrays (FPGAs) allowed the power electronics community to start investigating the viability of some sophisticated optimisation algorithms for facilitating the implementation of long-horizon MPC in real-time. In general, the following two strategies are the most popular.

***Multi-parametric programming*** solves the optimisation problem offline for all possible states. It uses the state vector as a parameter, which allows for computation of the control law and storage thereof in a look-up table for acquisition during the online process [30]. This methodology, known as *explicit* MPC [31], is computationally viable [32] but inflexible and ill-suited to address long-horizon MPC problems [8]. The main reasons are the offline computational burden and storage requirements associated with the complexity of the controller partition in higher dimensions.

***Mathematical programming*** techniques reduce the computational burden of the optimisation problem. The general approach is to formulate the optimisation problem as an integer quadratic program (IQP). Minimisation of the subsequent cost function then translates to solving an integer least-squares (ILS) problem. This problem, which

is also known as the closest vector problem (CVP) in lattice theory [33], is a well-known mathematical problem which can efficiently be solved. In communication applications, solving of the CVP is described as *decoding* [34], hence, the notion of a CVP *decoder* or decoding algorithm.

### 1.1.2.1 The closest vector problem

The general closest vector problem, as a function of the dimension, has been shown to be *non-deterministic polynomial-time hard* (NP-hard)<sup>1</sup> [36]. This implies that all *exact* CVP solutions have exponential *time-complexity*<sup>2</sup>, and it is highly unlikely that an algorithm will exist that can solve the problem in polynomial time. Practical systems usually employ some approximations, heuristics or combinations thereof as precondition to solving the problem. Approximate or heuristic methods obtain solutions faster and terminate in higher dimensions, but have been proven to be NP-hard for a certain degree of optimality/exactness [37]. Importantly, in [38] it was shown by simulation that *exact* solutions significantly outperform even the best heuristic methods. Algorithms that obtain the *exact* solution, according to [33], can be categorised in terms of their *space complexity*<sup>3</sup>:

***Polynomial-space algorithms*** are based on enumeration. Enumeration in its simplest form is an exhaustive search for the best integer combination of the basis vectors. Exhaustive search algorithms (ESAs) are computationally feasible only for small, low dimensional lattices, and are not practical for problems with thousands of sequences which arise from MPC formulations with prediction horizons of four or more [39, 40]. More sophisticated enumeration algorithms exist which follow two approaches, that differs mainly in the shape of the search region used for examining the lattice points. The Pohst strategy [41] uses a hyper-sphere, and the Kannan strategy [42] proposes a rectangular parallelepiped. Agrell(2002) [34] generalises with the statement, “the Pohst method is intended as a *practical tool* while the method of Kannan is intended as a *theoretical tool*”. Finke and Pohst(1985) [43] developed the well-known Sphere Decoding algorithm (SDA) which has been rediscovered and improved several times. Schnorr and Euchner(1994) [44]

<sup>1</sup>The “hard” in NP-hard implies that if an algorithm exists that can solve such a problem in polynomial-time, then all NP problems can be solved in polynomial time. [35]

<sup>2</sup>Time complexity quantifies the time taken by an algorithm to run as a function of the input length.

<sup>3</sup>Space complexity refers to the amount of memory required by an algorithm to execute as a function of the input length.

contributed by combining the SDA with the Babai nearest plane algorithm (heuristic), and proposed to examine the lattice points inside the hypersphere in a different order. With regard to power electronic systems in particular, the adapted SDA proposed by Geyer(2014) [39] has been used in a number of practical implementations.

**Exponential-space algorithms** have a better asymptotic running time than polynomial-space algorithms but with the added drawback that the solution requires *exponential space*. Ajtai, Kumar and Sivakumar(2001) [45] proposed the first of this kind, namely the *randomized sieve algorithm* (RSA). The technique, known as sieving, solves lattice problems exactly in running time of  $2^{O(n)}$ . Recently, Micciancio and Voulgaris(2013) [46] presented an algorithm, i.e. the Micciancio Voulgaris algorithm (MVA), which is deterministic and asymptotically faster than the randomised sieve algorithm. Although both the RSA and MVA are fast in solving lattice problems exactly, the exponential space requirement, as a result, has limited the application of these algorithms in practical implementations.

## 1.2 Objectives

The main objective of this thesis is to devise an alternative strategy that will solve the CVP problem relating to long-horizon FCS-MPC for multilevel inverters exactly in real time. To demonstrate the benefit of long horizons to power electronic systems of higher order, a plant involving a three-phase multilevel inverter that drives an induction machine via an intermediate *LC* filter is selected. Considering a linear system without any state constraints, the optimiser or decoding strategy resulting from this work should solve the optimisation problem underlying FCS-MPC and deliver an optimal control to the power electronic system within a single sample period. This includes steady-state and transient operating conditions. The proposed strategy will include aspects of offline preprocessing, online preconditioning of the CVP target, and an alternative decoding stage.

Partially preprocessing the solution offline should be efficient and result in reasonable storage space requirements. The online computational burden is split between two newly proposed algorithms. Preconditioning of the CVP target with a projection algorithm assists the decoding process, and is therefore expected to reduce the overall computational

effort, while maintaining optimality of the newly projected target. The proposed decoding algorithm, based on the SDA and tailored to the specific control set of a three-phase, three-level, neutral point clamped (NPC) inverter, should terminate and deliver the optimal control before the next sampling period commences.

## 1.3 Thesis outline

This thesis is organised into seven chapters with the content summarized as follow:

### 1.3.1 Chapter 1: Introduction

A brief overview of the control strategies in power electronics is covered with attention directed to the emerging field of MPC. The concept of long-horizon MPC and its associated advantages are mentioned. Computational complexity of the long-horizon FCS-MPC problem in particular is identified as a major obstacle for practical implementations. Associating the optimisation problem underlying FCS-MPC with the CVP in lattice theory leads to the introduction of various solvers or decoders. The objectives of this thesis are presented, of which the primary objective is to develop an alternative decoding strategy that can facilitate long-horizon FCS-MPC in real-time applications.

### 1.3.2 Chapter 2: Preliminaries

The general model predictive control strategy for power electronic systems is presented in terms of prediction, optimisation and the receding horizon policy. Formulation of the optimisation problem follows with inclusion of the cost function that conveys output variable tracking and switching cost. The optimisation problem is reformulated and presented in terms of the unconstrained minimum, which subsequently defines it as an integer least-squares problem. Physical systems relevant to this study and their respective mathematical models are noted and defined. The chapter concludes with the predictive control and performance parameters related to induction machines.

### 1.3.3 Chapter 3: Search space and solvers

The integer linear-squares problem is related to the closest vector problem in lattice theory which necessitates the discussion on lattices and their structures. Formulation of

the closest vector problem suggests that borders exist between lattice points, hence the concept of space partitioning and the basic Voronoi cell of a lattice are presented. The CVP of a truncated lattice is formulated with various solvers considered and analysed in terms of their expected computational complexities. Effects of lattice truncation and the need for target preconditioning is highlighted. This is followed by the proposal of an iterative projection algorithm and two approaches for maintaining optimality of the projection process.

### 1.3.4 Chapter 4: Performance estimation and development

This chapter is dedicated to the theoretical simulation and quantification of the computational efforts exerted by the decoding algorithms scrutinised in Chapter 3. A relatively simple PE system in the form of a three-phase NPC inverter driving an  $RL$ -load is modeled and subjected to FCS-MPC. The predictive controller calls on the newly proposed projection algorithm for target preconditioning and utilize the Micciancio Voulgaris algorithm for the final decoding stage. Performance indicators from the simulation are measured against those of the benchmark SDA. This allows for considering the respective advantages and formulation of an alternative approach for decoding.

### 1.3.5 Chapter 5: Alternative sphere decoder

Performance improvements observed in the MATLAB<sup>©</sup> simulations are contributed to block matrix operations that are favoured by modern processing hardware. The basic concept is discussed and subsequently proposed as an alteration to the conventional SDA. An alternative in the form of the sphere block decoding algorithm (SBDA) is presented, which is analysed in terms of computational complexity and MATLAB<sup>©</sup> simulations. Commitment to marry the MVA to a suitable power electronic application as an extension of the three-level NPC inverter is considered in the form of diode-clamped, multilevel inverters with a larger amount of voltage levels.

### 1.3.6 Chapter 6: Real-time verification

In this chapter, the real-time performance of the proposed decoding algorithm is studied for an induction machine (IM) drive with a three-phase NPC inverter that drives the motor via an intermediate  $LC$ -filter. The power electronic system is modeled and the

MPC problem is formulated accordingly. Various aspects that economise the optimisation process are considered; these include, pre-processing, computation of the unconstrained minimum and selection of the sampling frequency based on the orthogonality defect of the lattice basis. Practical considerations prelude the experimental setup and real-time simulation of the system. Steady-state operation and transients are invoked to evaluate the response of the controller and decoder termination time. Lastly, multilevel inverters with voltage levels more than three are considered for the IM drive. Comparative performance metrics are obtained for the SDA, SBDA and MVA respectively.

### **1.3.7 Chapter 7: Summary**

A summary of the contributions of this thesis is provided in this chapter, as well as a discussion of future research directions.



# Chapter 2

## Preliminaries

In this chapter, the theoretical background is covered as a basis for the work presented. First, the basic control problem is discussed, followed by the fundamentals of MPC. The optimisation problem underlying FCS-MPC is defined, with possible optimisers/decoders to the problem presented. Lastly, the physical power electronic systems and subsystems of relevance to this study are introduced.

### 2.1 The control problem

In the field of power electronics, the system or plant to be controlled typically consists of a discrete actuator, i.e. a power electronic converter and its load. Figure 2.1 shows a basic control structure. The general idea is for the controller to govern the system or plant in such a manner that its output follows a predetermined reference. This is achieved when the controller obtains measurements of the system's output variables  $\mathbf{y}$ , compares it to the reference values  $\mathbf{y}^*$  and then manipulates the input or control variable  $\mathbf{u}$  accordingly.

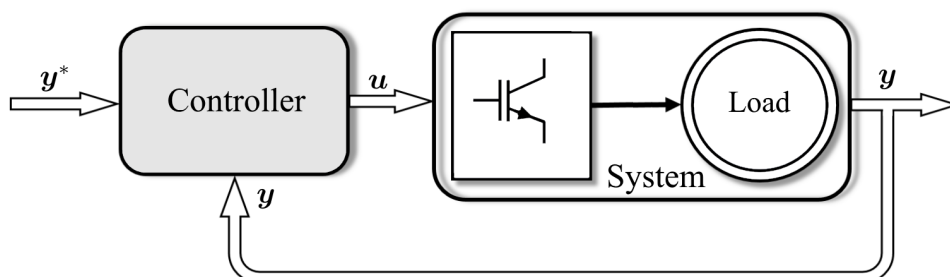


Figure 2.1: Typical control structure of a power electronics system.

In state-space form, the continuous-time differential equations for a linear time-invariant (LTI) system can be written as:

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{F}\mathbf{x}(t) + \mathbf{G}\mathbf{u}(t) \quad (2.1)$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) \quad (2.2)$$

with  $\mathbf{x} \in \mathbb{R}^{d_x}$  denoting the state vector,  $\mathbf{u} \in \mathbb{R}^{d_u}$  the input vector, and  $\mathbf{y} \in \mathbb{R}^{d_y}$  the output vector. The respective state, input and output matrices are represented by  $\mathbf{F} \in \mathbb{R}^{d_x \times d_x}$ ,  $\mathbf{G} \in \mathbb{R}^{d_x \times d_u}$  and  $\mathbf{C} \in \mathbb{R}^{d_y \times d_x}$ , where  $d_x$ ,  $d_u$  and  $d_y$  respectively denote the dimension of the state, input and output vector. Controllers typically operate at discrete time instants

$$t = kT_s, \quad k \in \mathbb{N} \triangleq \{0, 1, 2, \dots\} \quad (2.3)$$

where  $T_s$  is the sampling interval. Discretisation of the system continuous-time equations with  $T_s$  allows for the general formulation of the corresponding *discrete time model* in the state space

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) \quad (2.4)$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k). \quad (2.5)$$

For linear systems exact discretisation is possible and result in matrices

$$\mathbf{A} = e^{\mathbf{F}T_s} \quad \text{and} \quad \mathbf{B} = \int_0^{T_s} e^{\mathbf{F}\tau} d\tau \mathbf{G} \quad (2.6)$$

with  $\mathbf{C}$  remaining unaltered. If  $\mathbf{F}$  is non-singular then  $\mathbf{B}$  can be simplified to

$$\mathbf{B} = -\mathbf{F}^{-1} (\mathbf{I} - \mathbf{A}) \mathbf{G} \quad (2.7)$$

with  $\mathbf{I}$  the identity matrix of appropriate dimension. An exact solution of the continuous-time differential equations is not always obtainable for nonlinear systems. However, approximations thereof can be found with the Euler-forward, -backward or Runge-Kutta approximation methods.

## 2.2 Model predictive control

### 2.2.1 Fundamentals of MPC

Consider Figure 2.2, which presents a block diagram of the general model predictive control strategy for a power electronic system. At every sampling instant ( $k$ ) the current

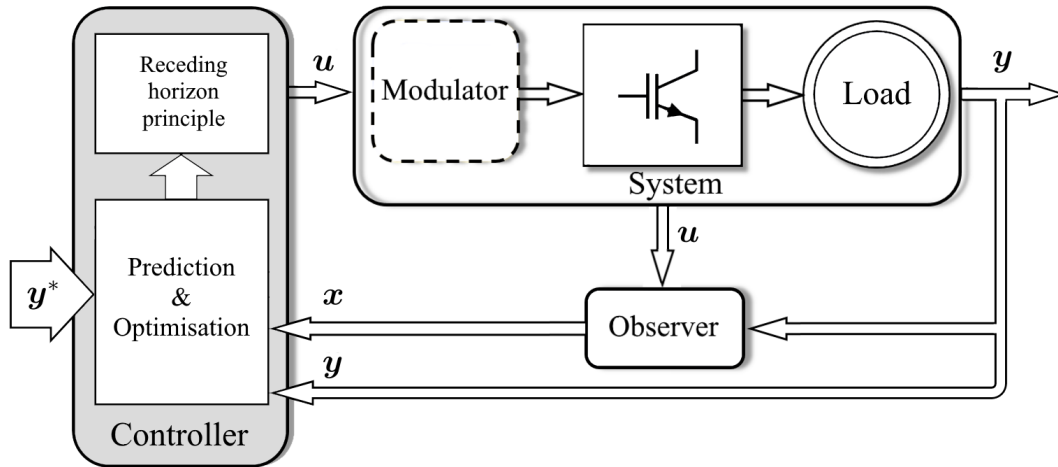


Figure 2.2: Typical MPC structure in the field of power electronics.

state  $\mathbf{x}(k)$  of the system is acquired and used as an initial state to predict the future behaviour of the system for a sequence of input variables over a finite prediction horizon of  $N$ -sampling intervals. The predictions are subjected to a predefined optimisation criterion, which in MPC is expressed as the minimisation of a cost function. The cost function conveys the system control objectives which can be weighed to penalise certain system behaviours. The sequence of input variables that results in a prediction with minimal cost is selected as the optimal input sequence  $\hat{\mathbf{u}}(k)$ , and the first action in this sequence is then used as input  $\mathbf{u}_{opt}(k)$  to the system. In cases where some components of the system state  $\mathbf{x}(k)$  cannot be measured, an observer is required to estimate these values.

As mentioned before, MPC can be subdivided into *continuous control set* and *finite control set* MPC. In CCS-MPC, an intermediate modulator is required to deliver the control signal to the converter with the system input  $\mathbf{u}_{opt}(k)$  constrained to a bounded continuous set  $\mathbf{U} \in \mathbb{R}^{d_u}$ , typically a voltage reference. FCS-MPC explicitly considers the discrete nature of the power electronic actuator, and delivers an optimum *switching state* for actuation. No modulator is required, and the input variable  $\mathbf{u}_{opt}(k)$  is constrained to a finite set  $\mathbf{U} \in \mathbb{Z}^{d_u}$  of switching combinations, defined by the converter topology in use. Although many variants of MPC controllers exist, all of them exhibit the same basic structure of *prediction*, *optimisation* and application of the *receding horizon principle*.

### 2.2.1.1 Cost function

In MPC, the system control objectives are conveyed in the *cost function*

$$J(\mathbf{x}(k), \tilde{\mathbf{u}}(k)) = \sum_{l=k}^{k+N-1} \Lambda(\mathbf{x}(l), \mathbf{u}(l)), \quad (2.8)$$

which is the sum of all the weighing functions  $\Lambda(\cdot, \cdot)$  or intermediate costs, accumulated over the finite prediction horizon. The cost function evaluates the effect that a predicted input sequence

$$\tilde{\mathbf{u}}(k) = \left[ \mathbf{u}^T(k) \quad \mathbf{u}^T(k+1) \dots \mathbf{u}^T(k+N-1) \right]^T \in \mathcal{U}^N, \quad (2.9)$$

starting from an initial system state  $\mathbf{x}(k)$ , will have on the system behaviour. Prediction of the system behaviour is obtained from the discrete-time system model and its evolution over the prediction horizon, i.e.

$$\mathbf{x}(l+1) = \mathbf{A}\mathbf{x}(l) + \mathbf{B}\mathbf{u}(l) \quad (2.10a)$$

$$\text{with } l = k + n - 1 \quad \text{and} \quad n = 1, 2, \dots, N. \quad (2.10b)$$

### 2.2.1.2 Optimisation

Inclusion of the system model (2.10) and relevant input constraints  $\mathcal{U}$ , into the minimisation process allows for stating the optimisation problem

$$\hat{\mathbf{u}}(k) = \arg \min_{\tilde{\mathbf{u}}(k)} J(\mathbf{x}(k), \tilde{\mathbf{u}}(k)) \quad (2.11a)$$

$$\text{subject to } \mathbf{x}(l+1) = \mathbf{A}\mathbf{x}(l) + \mathbf{B}\mathbf{u}(l) \quad (2.11b)$$

$$\mathbf{y}(l+1) = \mathbf{C}\mathbf{x}(l+1) \quad (2.11c)$$

$$\mathbf{u}(l) \in \mathcal{U} \quad (2.11d)$$

$$\forall l = k, k+1, \dots, k+N-1. \quad (2.11e)$$

The solution to the optimisation problem is an optimal control vector or sequence of input variables

$$\hat{\mathbf{u}}(k) \in \mathbb{U} = \mathcal{U}^N. \quad (2.12)$$

If CCS-MPC with  $\mathcal{U} \in \mathbb{R}^{d_u}$  is considered, the sequence of input variables is real-valued and solving the optimisation (2.11) will require a quadratic program (QP). In contrast will FCS-MPC require an integer program (IP) due to the integer-valued control set  $\mathcal{U} \in \mathbb{Z}^{d_u}$ .

### 2.2.1.3 Receding horizon strategy

When prediction horizons of more than one are used, the receding horizon principle [47] is called upon. This is the process whereby only the first action  $\mathbf{u}_{opt}(k)$  of the optimal input sequence is used to actuate the system. With  $\hat{\mathbf{u}}(k)$ , an open-loop constrained optimal input sequence that depends on the state measurement  $\mathbf{x}(k)$ , this procedure closes the control loop and introduces feedback into the MPC law. MPC that uses larger values for the horizon length  $N$ , in general, will provide better performance [39, 48, 32]. For CCS-MPC in particular, a large enough prediction horizon will marginalise the effect of  $\mathbf{u}_{opt}(k)$  on  $\mathbf{x}(l)$  for  $l > k + N$ , and MPC will approximate the performance of an infinite horizon optimal controller [49].

## 2.3 The optimisation problem underlying FCS-MPC

Unfortunately, the prospect of MPC with long prediction horizons comes with a price; more computations are required to solve the underlying optimisation problem, which in a real-time implementation becomes problematic. Especially in FCS-MPC that contain integer values, the computational demand exponentially increases with extending the prediction horizon. Since this work is mainly concerned with FCS-MPC, any reference to  $\mathbf{u} \in \mathcal{U}$  from this point forward will imply an *integer-valued* vector belonging to a *finite* set.

### 2.3.1 Formulation of the cost function

In a power converter application, the fundamental trade-off between harmonic distortion in the output and internal switching losses is of main concern. These issues relate directly to the two conflicting control objectives of reference tracking and converter switching frequency. Including the aforementioned objectives in a general cost function, formulated in quadratic form gives

$$J = \sum_{l=k}^{k+N-1} \|\mathbf{y}_e(l+1)\|_{\Lambda}^2 + \lambda_u \|\mathbf{u}_e(l)\|^2. \quad (2.13)$$

The first term

$$\|\mathbf{y}_e(l+1)\|_{\Lambda}^2 = [\mathbf{y}_e(l+1)]^T \Lambda [\mathbf{y}_e(l+1)], \quad (2.14)$$

with

$$\mathbf{y}_e(l+1) = \mathbf{y}^*(l+1) - \mathbf{y}(l+1) \quad (2.15)$$

signifies the tracking error i.e. the difference between the reference  $\mathbf{y}^*$  and output  $\mathbf{y}$ , adjusted with a weighing or penalty matrix  $\mathbf{\Lambda}$ . Note that obtaining the squared norm of the vector norm of  $\mathbf{y}_e(l+1)$ , the weighing matrix  $\mathbf{\Lambda}$  must (by definition) be positive semi-definite and symmetric. The second term in (2.18) quantifies the switching effort

$$\mathbf{u}_e(l) = \mathbf{u}(l) - \mathbf{u}(l-1) \quad (2.16)$$

with  $\lambda_u$  denoting the switching weight, also referred to as a tuning parameter. Varying  $\lambda_u$  adjusts the switching cost weight and hence the trade-off between reference tracking accuracy and converter switching frequency.

### 2.3.1.1 Cost function in vector form

By successively applying (2.4), the state vector at time-steps  $(k+n)$ ,  $n = 1, 2, \dots, N$  can be represented as a function of the state vector at time-step  $(k)$  and the control sequence from time-step  $(k)$  to  $(k+n-1)$ , i.e.

$$\mathbf{x}(k+n) = \mathbf{A}^n \mathbf{x}(k) + \mathbf{A}^{n-1} \mathbf{B} \mathbf{u}(k) + \dots + \mathbf{A}^0 \mathbf{B} \mathbf{u}(k+n-1). \quad (2.17)$$

Substituting (2.17) into (2.5) defines the output sequence at time-step  $k+n$  as

$$\mathbf{y}(k+n) = \mathbf{C} \mathbf{A}^n \mathbf{x}(k) + \mathbf{C} \mathbf{A}^{n-1} \mathbf{B} \mathbf{u}(k) + \dots + \mathbf{C} \mathbf{A}^0 \mathbf{B} \mathbf{u}(k+n-1). \quad (2.18)$$

Introducing the sequence of output variables as the vector

$$\tilde{\mathbf{y}}(k) = \begin{bmatrix} \mathbf{y}^T(k+1) & \mathbf{y}^T(k+2) & \dots & \mathbf{y}^T(k+N) \end{bmatrix}^T \in \mathbb{R}^{Nd_y}, \quad (2.19)$$

allows for writing it in matrix notation

$$\tilde{\mathbf{y}}(k) = \mathbf{\Gamma} \mathbf{x}(k) + \mathbf{\Upsilon} \tilde{\mathbf{u}}(k), \quad (2.20)$$

where

$$\mathbf{\Gamma} = \begin{bmatrix} \mathbf{CA} \\ \mathbf{CA}^2 \\ \mathbf{CA}^3 \\ \vdots \\ \mathbf{CA}^N \end{bmatrix} \quad \text{and} \quad \mathbf{\Upsilon} = \begin{bmatrix} \mathbf{CB} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{CAB} & \mathbf{CB} & \cdots & \mathbf{0} \\ \mathbf{CA}^2\mathbf{B} & \mathbf{CAB} & \mathbf{CB} & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{CA}^{N-1}\mathbf{B} & \mathbf{CA}^{N-2}\mathbf{B} & \cdots & \mathbf{CB} \end{bmatrix}.$$

Including this dynamic model in the cost function and introducing the block diagonal matrix  $\tilde{\mathbf{\Lambda}} = \text{diag}(\mathbf{\Lambda}, \dots, \mathbf{\Lambda})$  yields

$$J = \|\mathbf{\Gamma}\mathbf{x}(k) + \mathbf{\Upsilon}\tilde{\mathbf{u}}(k) - \tilde{\mathbf{y}}^*(k)\|_{\tilde{\mathbf{\Lambda}}}^2 + \lambda_u \|\mathbf{S}\tilde{\mathbf{u}}(k) - \mathbf{E}\mathbf{u}(k-1)\|^2 \quad (2.21)$$

where  $\tilde{\mathbf{y}}^*$  denotes the output reference vector

$$\tilde{\mathbf{y}}^*(k) = \begin{bmatrix} \mathbf{y}^{*T}(k+1) & \mathbf{y}^{*T}(k+2) & \cdots & \mathbf{y}^{*T}(k+N) \end{bmatrix}^T. \quad (2.22)$$

The auxiliary matrices

$$\mathbf{S} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ -\mathbf{I} & \mathbf{I} & \cdots & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} & \mathbf{I} & \mathbf{0} \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{I} \end{bmatrix} \quad \text{and} \quad \mathbf{E} = \begin{bmatrix} \mathbf{I} \\ \mathbf{0} \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}$$

are of appropriate dimension and constructed from the identity matrix  $\mathbf{I}$  and zero matrix  $\mathbf{0}$ . Expanding (2.21) and rearranging the terms allow for the cost function to be written in the form

$$J = \|\mathbf{\Gamma}\mathbf{x}(k) - \tilde{\mathbf{y}}^*(k)\|_{\tilde{\mathbf{\Lambda}}}^2 + \lambda_u \|\mathbf{E}\mathbf{u}(k-1)\|^2 + 2[\mathbf{\Gamma}\mathbf{x}(k) - \tilde{\mathbf{y}}^*(k)]^T \tilde{\mathbf{\Lambda}}\mathbf{\Upsilon}\tilde{\mathbf{u}}(k) - 2\lambda_u [\mathbf{E}\mathbf{u}(k-1)]^T \mathbf{S}\tilde{\mathbf{u}}(k) + [\tilde{\mathbf{u}}(k)]^T \left( \mathbf{\Upsilon}^T \tilde{\mathbf{\Lambda}} \mathbf{\Upsilon} + \lambda_u \mathbf{S}^T \mathbf{S} \right) \tilde{\mathbf{u}}(k), \quad (2.23)$$

or compact form

$$J = \theta(k) + 2 [\Theta(k)]^T \tilde{\mathbf{u}}(k) + [\tilde{\mathbf{u}}(k)]^T \mathbf{Q} \tilde{\mathbf{u}}(k) \quad (2.24)$$

with

$$\theta(k) = \|\Gamma \mathbf{x}(k) - \tilde{\mathbf{y}}^*(k)\|_{\tilde{\Lambda}}^2 + \lambda_u \|\mathbf{E} \mathbf{u}(k-1)\|^2 \quad (2.25)$$

$$[\Theta(k)]^T = [\Gamma \mathbf{x}(k) - \tilde{\mathbf{y}}^*(k)]^T \tilde{\Lambda} \Upsilon - \lambda_u [\mathbf{E} \mathbf{u}(k-1)]^T \mathbf{S} \quad (2.26)$$

$$\mathbf{Q} = \Upsilon^T \tilde{\Lambda} \Upsilon + \lambda_u \mathbf{S}^T \mathbf{S}. \quad (2.27)$$

$\mathbf{Q}$  is per definition symmetric and positive definite for  $\lambda_u \geq 0$ , allowing for the last term in (2.24) to be written as

$$[\tilde{\mathbf{u}}(k)]^T \mathbf{Q} \tilde{\mathbf{u}}(k) = \|\tilde{\mathbf{u}}(k)\|_{\mathbf{Q}}^2, \quad (2.28)$$

updating the cost function to its final form

$$J = \|\tilde{\mathbf{u}}(k)\|_{\mathbf{Q}}^2 + 2 [\tilde{\mathbf{u}}(k)]^T \Theta(k) + \theta(k). \quad (2.29)$$

### 2.3.1.2 Cost function in terms of the unconstrained minimum

Completing the square of

$$[\tilde{\mathbf{u}}(k) + \mathbf{Q}^{-1} \Theta(k)]^T \mathbf{Q} [\tilde{\mathbf{u}}(k) + \mathbf{Q}^{-1} \Theta(k)] \quad (2.30a)$$

$$= [\tilde{\mathbf{u}}(k)]^T \mathbf{Q} \tilde{\mathbf{u}}(k) + 2 [\tilde{\mathbf{u}}(k)]^T \Theta(k) + [\Theta(k)]^T \mathbf{Q}^{-T} \Theta(k) \quad (2.30b)$$

$$= \|\tilde{\mathbf{u}}(k)\|_{\mathbf{Q}}^2 + 2 [\tilde{\mathbf{u}}(k)]^T \Theta(k) + \|\Theta(k)\|_{\mathbf{Q}^{-T}}^2. \quad (2.30c)$$

and expanding  $J$  from (2.29) with (2.30c) gives

$$J = \|\tilde{\mathbf{u}}(k)\|_{\mathbf{Q}}^2 + 2 [\tilde{\mathbf{u}}(k)]^T \Theta(k) + \|\Theta(k)\|_{\mathbf{Q}^{-T}}^2 - \|\Theta(k)\|_{\mathbf{Q}^{-T}}^2 + \theta(k) \quad (2.31a)$$

$$= [\tilde{\mathbf{u}}(k) + \mathbf{Q}^{-1} \Theta(k)]^T \mathbf{Q} [\tilde{\mathbf{u}}(k) + \mathbf{Q}^{-1} \Theta(k)] - \|\Theta(k)\|_{\mathbf{Q}^{-T}}^2 + \theta(k). \quad (2.31b)$$

In (2.31b) the last two terms are functions of  $\mathbf{x}(k)$  and  $\mathbf{u}(k-1)$ , thus independent of the optimisation variable  $\tilde{\mathbf{u}}(k)$ . This implies that both terms will remain constant during the optimisation process and have no influence on the expected outcome. Omitting these terms for the purpose of *minimisation* results in a *reduced cost function*

$$\tilde{J} = [\tilde{\mathbf{u}}(k) + \mathbf{Q}^{-1} \Theta(k)]^T \mathbf{Q} [\tilde{\mathbf{u}}(k) + \mathbf{Q}^{-1} \Theta(k)] \quad (2.32a)$$

$$= \|\tilde{\mathbf{u}}(k) + \mathbf{Q}^{-1} \Theta(k)\|_{\mathbf{Q}}^2. \quad (2.32b)$$



The the second term in (2.32b) is defined as the unconstrained minimum

$$\tilde{\mathbf{u}}_{unc}(k) = -\mathbf{Q}^{-1}\Theta(k). \quad (2.33)$$

If  $\tilde{\mathbf{u}}(k)$  was real-valued, then  $\tilde{\mathbf{u}}_{unc}(k)$  would minimise the cost function and result in  $\tilde{J} = 0$ . Obtaining the *Cholesky decomposition*<sup>4</sup> for  $\mathbf{Q}^{-1}$  gives

$$\mathbf{Q}^{-1} = \mathbf{H}^{-1}\mathbf{H}^{-T} \quad (2.34a)$$

$$\mathbf{Q} = \mathbf{H}^T\mathbf{H}. \quad (2.34b)$$

The Cholesky decomposition is chosen in this work mainly for the purpose of reducing the computational complexity of the optimisation process. Noteworthy, decompositions which address other issues also exist. For example, Aguilera(2014) [51] proposed the decomposition  $\mathbf{H} = \mathbf{Q}^{1/2}$ , which ensures closed-loop stability. Henceforward, substituting (2.33) and (2.34b) in (2.32a) allows for rewriting the reduced cost function as

$$\tilde{J} = [\tilde{\mathbf{u}}(k) - \tilde{\mathbf{u}}_{unc}(k)]^T \mathbf{H}^T \mathbf{H} [\tilde{\mathbf{u}}(k) - \tilde{\mathbf{u}}_{unc}(k)] \quad (2.35a)$$

$$= [\mathbf{H}\tilde{\mathbf{u}}(k) - \mathbf{H}\tilde{\mathbf{u}}_{unc}(k)]^T [\mathbf{H}\tilde{\mathbf{u}}(k) - \mathbf{H}\tilde{\mathbf{u}}_{unc}(k)] \quad (2.35b)$$

$$= \|\mathbf{H}\tilde{\mathbf{u}}(k) - \mathbf{H}\tilde{\mathbf{u}}_{unc}(k)\|^2. \quad (2.35c)$$

### 2.3.2 Optimisation problem

Consideration of the reduced cost function (2.35c), allows for defining the optimisation problem related to FCS-MPC with reference tracking as

$$\hat{\mathbf{u}}(k) = \arg \min_{\tilde{\mathbf{u}}(k)} \|\mathbf{H}\tilde{\mathbf{u}}(k) - \mathbf{H}\tilde{\mathbf{u}}_{unc}(k)\|^2 \quad (2.36a)$$

$$\text{subject to } \tilde{\mathbf{u}}(k) \in \mathbb{U} \quad (2.36b)$$

$$\text{with} \quad (2.36c)$$

$$\tilde{\Lambda} = \text{diag}(\Lambda, \dots, \Lambda) \quad (2.36d)$$

$$\mathbf{Q} = \Upsilon^T \tilde{\Lambda} \Upsilon + \lambda_u \mathbf{S}^T \mathbf{S} \quad (2.36e)$$

$$\mathbf{H}^T \mathbf{H} = \mathbf{Q} \quad (2.36f)$$

$$[\Theta(k)]^T = [\Gamma \mathbf{x}(k) - \tilde{\mathbf{y}}^*(k)]^T \tilde{\Lambda} \Upsilon - \lambda_u [\mathbf{E} \mathbf{u}(k-1)]^T \mathbf{S} \quad (2.36g)$$

$$\tilde{\mathbf{u}}_{unc}(k) = -\mathbf{Q}^{-1}\Theta(k) \quad (2.36h)$$

$$\mathbb{U} = \mathcal{U}^N \subset \mathbb{Z}^{N d_u}. \quad (2.36i)$$

---

<sup>4</sup>Every real-valued symmetric positive-definite matrix  $\mathbf{A}$  has a unique *Cholesky decomposition* that results in a unique lower triangular matrix  $\mathbf{B}$  with strictly real and positive diagonal entries such that  $\mathbf{A} = \mathbf{B}\mathbf{B}^T$  [50, p.143].

The optimisation problem (2.36) resembles an *integer quadratic program*, truncated with the feasible set  $\mathbb{U}$ . The process of minimising of the cost (2.36a) subject to (2.36b) is a so-called *integer least-squares* (ILS) problem. Solving the ILS problem i.e. obtaining  $\hat{\mathbf{u}}$ , geometrically translates to a search in the  $\mathbf{H}$ -coordinate space for a vector from the set  $\mathbf{H}\tilde{\mathbf{u}}$ , with minimum Euclidean distance to the transformed unconstrained solution  $\mathbf{H}\tilde{\mathbf{u}}_{unc}$ . This relates to the traditional *nearest neighbour problem* (NNP) which is also known as the *closest vector problem* (CVP) in lattice theory. Dimensionality of the  $\mathbf{H}$ -coordinate space as a function of the control vector dimension  $d_u$ , and the prediction horizon  $N$  is introduced as

$$d = Nd_u. \quad (2.37)$$

## 2.4 Physical systems

### 2.4.1 Three phase systems

The instantaneous phase voltages of a balanced alternating three-phase system at time  $t$  are defined by

$$\begin{aligned} v_a(t) &= \sqrt{2}V_R \sin(\omega t), \\ v_b(t) &= \sqrt{2}V_R \sin\left(\omega t - \frac{2}{3}\pi\right), \\ v_c(t) &= \sqrt{2}V_R \sin\left(\omega t - \frac{4}{3}\pi\right), \end{aligned}$$

with  $V_R$  the root-mean-square (rms) value of the rated phase voltage, and  $\omega = 2\pi f$  denoting the angular frequency at an alternating frequency  $f$ .

#### 2.4.1.1 Stationary reference frame

To simplify the modeling of three-phase circuits, the three-phase, three-dimensional  $abc$  system variables are transformed to the orthogonal  $\alpha\beta 0$  reference frame by means of the

Clarke-transformation matrix [52]

$$\begin{bmatrix} \alpha \\ \beta \\ 0 \end{bmatrix} = \frac{2}{3} \begin{bmatrix} 1 & -1/2 & -1/2 \\ 0 & \sqrt{3}/2 & -\sqrt{3}/2 \\ 1/2 & 1/2 & 1/2 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \end{bmatrix}.$$

The transformation can be considered as the writing the  $abc$  quantities in an orthogonal coordinate system using basis vectors that point in the directions of the  $a$ -,  $b$ - and  $c$ -axis. If the neutral point in a balanced three-phase system is not grounded, the 0 component in the  $\alpha\beta 0$  frame is omitted. This results in a simplified two-phase, two-dimensional representation of the three-phase system. In this case the Clarke transformation is reduced to

$$\mathbf{K} = \frac{2}{3} \begin{bmatrix} 1 & -1/2 & -1/2 \\ 0 & \sqrt{3}/2 & -\sqrt{3}/2 \end{bmatrix}, \quad (2.38)$$

with pseudo-inverse

$$\mathbf{K}^{-1} = \begin{bmatrix} 1 & 0 \\ -1/2 & \sqrt{3}/2 \\ -1/2 & -\sqrt{3}/2 \end{bmatrix}. \quad (2.39)$$

#### 2.4.1.2 Rotating reference frame

Generalisation of the stationary orthogonal reference frame to a rotating orthogonal reference frame results in a direct ( $d$ ), quadrature ( $q$ ) and zero (0) axis. The  $q$ -axis precedes the  $d$ -axis in rotation with the angular position of the latter defined as the angle to the  $a$ -axis of the three-phase system. Transformation from the stationary  $\alpha\beta$  coordinates to the rotating  $dq$  coordinates and *vice versa* are obtained through

$$\begin{bmatrix} d \\ q \end{bmatrix} = \mathbf{R} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \mathbf{R}^T \begin{bmatrix} d \\ q \end{bmatrix}$$

with the rotation matrix

$$\mathbf{R} = \begin{bmatrix} \cos\psi & \sin\psi \\ -\sin\psi & \cos\psi \end{bmatrix}. \quad (2.40)$$

## 2.5 Mathematical sub-system models

This section introduces the physical subsystems that constitute the general power electronic system or plant which is to be controlled. A typical power electronic system consists of linear circuit elements (inductors, capacitors, and resistors) and switching circuit elements which may either be controlled or uncontrolled semiconductor devices. For each switching combination, the power electronic system responds differently and can be described by linear differential equations in time. In cases where saturation effects, delays, and safety constraints are neglected, power electronic systems constitute switched *linear* systems. Where machine variables such as the electromagnetic torque or stator flux magnitude are directly controlled, non-linearities arise and the power electronic system then represents switched *nonlinear* systems. First, the selected multilevel inverter topologies are described, followed by the different loads considered in this thesis. Mathematical models of linear circuit elements are presented, as well as the dynamic model of an induction machine.

### 2.5.1 Multilevel inverters

In many drive applications, multilevel inverters are preferred to conventional two-level inverters [53]. Multilevel inverters are used because they provide higher output voltages and thus higher power [54, 55, 56]. Hence, the current trend towards multilevel inverters that outputs high-quality power with high efficiency.

#### 2.5.1.1 Diode-clamped inverter

The diode-clamped multilevel inverter proposed by [57] in the early 1980s was named the neutral point converter (NPC). It delivers a voltage waveform with three distinctive voltage levels, which allows for the semiconductor switches to be commutated at half the dc-link voltage. This trait makes it an attractive choice for medium voltage applications

[58] and lead to the NPC becoming a popular inverter topology which is currently employed by all major drive companies [5]. The single-phase topology of an NPC inverter, shown in Figure 2.3, consists of two capacitors on the DC bus which enable delivery of a phase voltage with three levels.

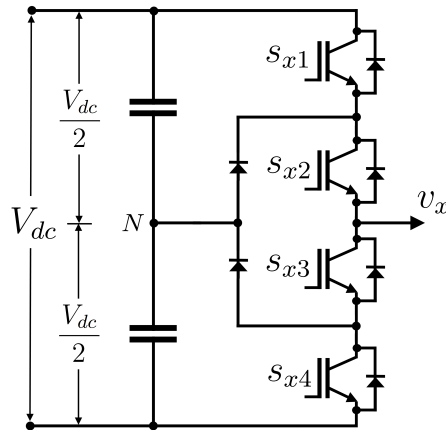


Figure 2.3: Neutral-point-clamped inverter topology.

Assuming the capacitor voltages to be regulated and constant at  $V_{dc}/2$ , an inverter leg  $x$  can deliver a voltage

$$v_x = \frac{V_{dc}}{2} u_x, \quad u_x \in \mathcal{U} \subset \mathbb{Z}, \quad (2.41)$$

with respect to the dc-link midpoint  $N$ . The integer value  $u_x$  represents the state of the inverter leg and  $\mathcal{U}$ , the set of possible leg states

$$\mathcal{U} = \pm \{-1 \ 0 \ 1\}. \quad (2.42)$$

To effect the output voltage levels commanded by the leg state  $u_x$ , the semiconductor switches in the inverter leg are actuated according to Table 2.1.

Table 2.1: NPC inverter leg state, switch states and output voltage level.

Leg state $u_x$	Switch states $s_{x1} \ s_{x2} \ s_{x3} \ s_{x4}$	Phase voltage $v_x$
1	on on off off	$+0.5V_{dc}$
0	off on on off	0
-1	off off on on	$-0.5V_{dc}$

### 2.5.1.2 Cascaded H-bridge inverter

Cascade multilevel inverters synthesize a multilevel voltage by connecting a number of single-phase H-bridge cells ( $C$ ) in series. A single cell ( $C = 1$ ), shown in Figure 2.4a can output ( $l_v = 2C + 1$ ) voltage levels, with the switch actions listed in Table 2.2.

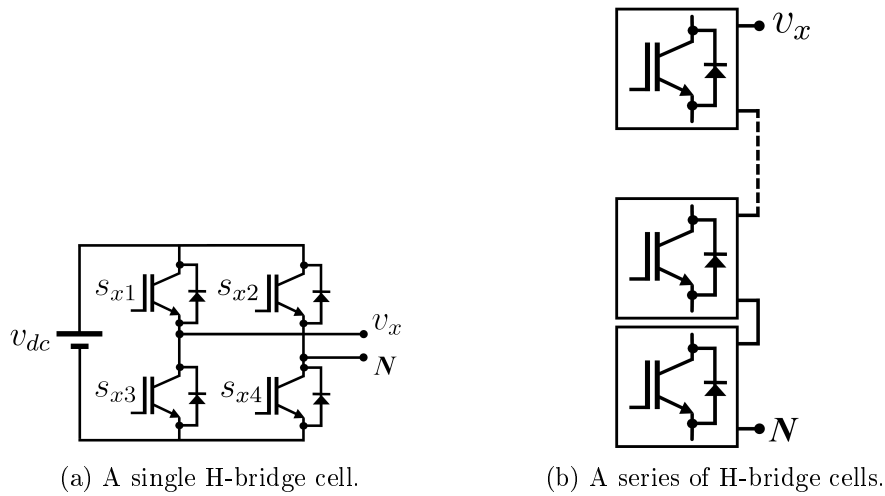


Figure 2.4: Cascaded H-bridge inverter topology.

Table 2.2: Cascaded H-bridge inverter cell leg state, switch states and output voltage level.

Leg state $u_x$	Switch states $s_{x1} \ s_{x2} \ s_{x3} \ s_{x4}$	Phase voltage $v_x$
1	on off off on	$+v_{dc}$
0	off off off off	0
-1	off on on off	$-v_{dc}$

The integer value  $u_x \in \mathcal{U}$ , representing the state of the inverter leg, defines the output voltage as

$$v_x = v_{dc}u_x, \quad u_x \in \mathcal{U}, \quad (2.43)$$

with respect to the neutral point  $N$ . For a CHB inverter leg that consists of a number of  $C$ -cells (Fig. 2.4b), and outputs  $l_v = 2C + 1$  levels, the set of possible leg states is defined

by

$$\mathcal{U} = \pm \left\{ \frac{(l_v - n)}{2}, n = 1, 3, \dots, l_v \right\}. \quad (2.44)$$

### 2.5.1.3 Three-phase topology

The single-phase topologies introduced above is used as building blocks for their respective three-phase counterparts. A three-phase NPC inverter is merely three single-phase inverter arms cascaded onto the same dc-rails. The neutral point potential  $v_N$ , which is defined as the difference between the voltages over the upper and lower dc-link capacitors floats. In this work the neutral point potential is assumed to be zero.

To constitute a three-phase CHB inverter, the neutral points of three CHB inverter legs are joined together and the respective outputs connected to the three-phase load. In both topologies the three inverter legs deliver a phase voltage  $v_x, x \in \{a, b, c\}$  which is commanded by the relevant leg state value  $u_x \in \mathcal{U}$ . Combining the three leg states lead to the introduction of a three-dimensional *control vector*

$$\mathbf{u}_{abc} = [u_a \ u_b \ u_c]^T \in \mathbb{Z}^{d_u}, \quad (2.45)$$

where  $d_u = 3$  denotes the dimension of the control vector, i.e. number of phases. All combinations of the control vector constitute the inverter *control set*

$$\mathcal{U} = \mathcal{U}^{d_u}, \quad (2.46)$$

with  $\mathcal{U}^{d_u}$  being the  $d_u$ -times Cartesian product of the set  $\mathcal{U}$ . Hence the cardinality of the control set

$$\#\mathcal{U} = (\#\mathcal{U})^{d_u}. \quad (2.47)$$

Transformation of this set with the Clarke-transformation (2.38) to the stationary orthogonal  $\alpha\beta$  reference frame results in the associated *voltage vectors*

$$\mathbf{u}_{\alpha\beta} = \mathbf{K}\mathbf{u}_{abc}. \quad (2.48)$$

For example, Figure 2.5 shows the voltage vectors  $\mathbf{u}_{\alpha\beta}$  resulting from the ( $\#\mathcal{U} = 27$ )-control vectors of a three-phase, three-level inverter. Note that some redundancy is present in that some control vectors effect the same voltage vectors. Certain strategies choose to ignore the redundant switch options, but the MPC approach considers all of them to deliver an optimal control vector that minimises the converter switching losses.

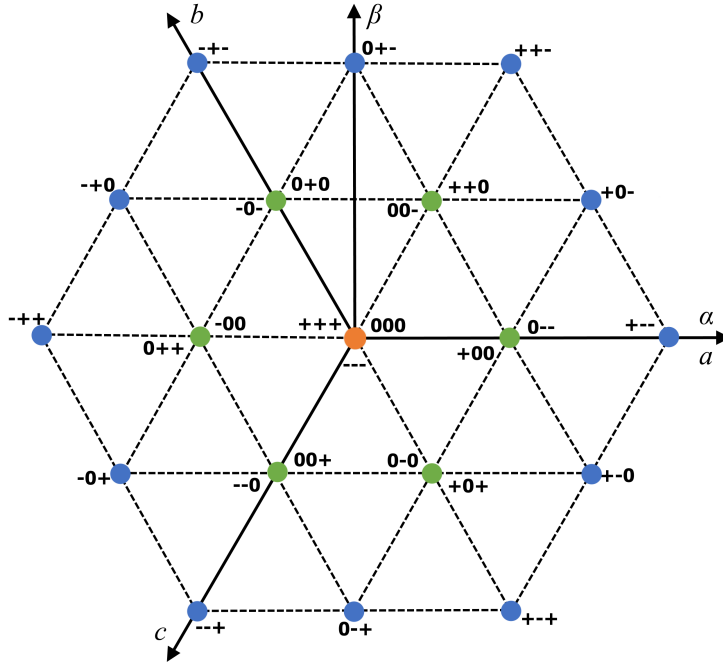


Figure 2.5: Voltage space vectors generated by a three-level NPC inverter in the  $\alpha\beta$  plane along with their congruent three-phase switch positions ("+" refers to "1" and "-" to "-1").

#### 2.5.1.4 Switching constraints and switching frequency

To prevent a short-circuit between the upper and the lower dc-link rails of an NPC inverter, switching between +1 and -1 in a single inverter leg is prohibited. Switching is therefore restricted to only one step up or one step down per control period ( $k$ ), as prescribed by the constraint

$$\|\mathbf{u}_{abc}(k) - \mathbf{u}_{abc}(k-1)\|_{\infty} \leq 1. \quad (2.49)$$

Control of an inverter by means of MPC results in a variable switching frequency. To obtain a measure of the switching losses incurred by the converter, the average switching frequency  $f_{sw}$  can be considered [5]. This frequency is found by recording the number of switching cycles per time interval where a cycle is defined as two transitions, i.e. *on* and *off*. In a three-phase NPC inverter, the inverter legs are controlled by the vector  $\mathbf{u}_{abc}$  which allows for defining the average switching frequency per inverter leg

$$f_{swl} = \lim_{n \rightarrow \infty} \frac{1}{6nT_s} \sum_{k=0}^{n-1} \|\mathbf{u}_{abc}(k) - \mathbf{u}_{abc}(k-1)\|_1. \quad (2.50)$$

The number of samples and the sampling period are denoted by  $n$  and  $T_s$  respectively. In order for an inverter leg with  $n_{sw}$ -switches to effect a *valid* transition, two switches



are engaged (see Table 2.1 and 2.2). Hence, the *average switching frequency* per semiconductor device is

$$f_{sw} = \frac{2f_{swl}}{n_{sw}}. \quad (2.51)$$

## 2.5.2 Linear circuit elements

Figures 2.6a and 2.6b respectively show the equivalent circuit diagrams of a symmetrical three-phase resistive-inductive ( $RL$ ) load, and inductive-capacitive ( $LC$ ) low pass filter. The input voltage and current to both circuits in the  $\alpha\beta$  coordinate system are denoted by  $\mathbf{v}_i = [v_{i\alpha} \ v_{i\beta}]^T$  and  $\mathbf{i}_i = [i_{i\alpha} \ i_{i\beta}]^T$ , with the output voltage and current of the filter circuit denoted by  $\mathbf{v}_s = [v_{s\alpha} \ v_{s\beta}]^T$  and  $\mathbf{i}_s = [i_{s\alpha} \ i_{s\beta}]^T$ .

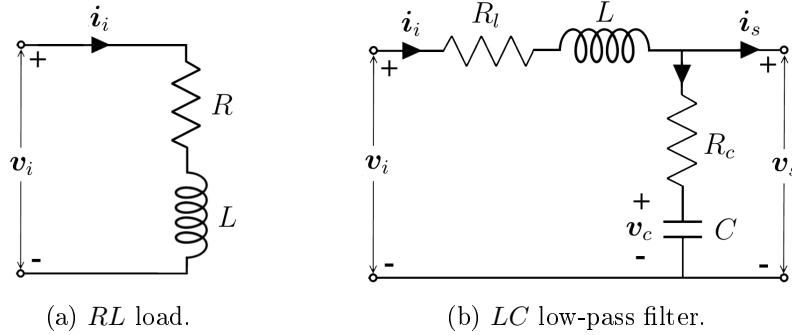


Figure 2.6: Linear circuits.

The differential equation of the  $RL$  circuit defined in state-space form is

$$\frac{d\mathbf{i}_i}{dt} = -\frac{R}{L}\mathbf{i}_i + \frac{1}{L}\mathbf{v}_i. \quad (2.52)$$

Selecting the state variables for the  $LC$  filter circuit as  $\mathbf{i}_i$  and  $\mathbf{v}_c$  allows for defining the differential state-space equations of the filter as

$$\frac{d\mathbf{i}_i}{dt} = \frac{1}{L}(\mathbf{v}_i - R_l\mathbf{i}_i - \mathbf{v}_s) \quad (2.53a)$$

$$\frac{d\mathbf{v}_c}{dt} = \frac{1}{C}(\mathbf{i}_i - \mathbf{i}_s) \quad (2.53b)$$

with the filtered voltage

$$\mathbf{v}_s = \mathbf{v}_c + R_c(\mathbf{i}_i - \mathbf{i}_s). \quad (2.54)$$

### 2.5.3 Induction machine

The stator windings of a squirrel cage induction machine are supplied by a three-phase system of voltages at an electrical angular frequency of  $\omega_s = 2\pi f_s$ , with  $f_s$  the frequency of the applied voltage. Three-phase currents are set up in the respective stator windings and generate magnetic fluxes, which combine to create a rotating magnetic field (RMF) in the air gap. The angular speed of the RMF, also referred to as synchronous speed, is defined by  $\Omega_s = \omega_s/p$ , where  $p$  denotes the number of pole pairs that make up the stator. The RMF induces a voltage in the rotor bars and set up currents, which in turn generates a magnetic field rotating with an electrical speed of  $\omega_r$ . This magnetic field interacts with the RMF and results in a force that drives the motor with some developed electromagnetic torque  $T_e$ . The torque is proportional to the flux density and rotor bar current. For the motor to operate, the mechanical rotor speed  $\Omega_r = \omega_r/p$  must be less than the synchronous speed  $\Omega_s$ . This is necessary, otherwise the magnetic field will not move relative to the rotor conductors, prohibiting the induction of currents therein. The difference between the speed of the rotor and that of the RMF in the stator is called the slip  $s$ , and is defined as,

$$s = \frac{\Omega_s - \Omega_r}{\Omega_s}. \quad (2.55)$$

#### 2.5.3.1 Dynamic model of an induction machine

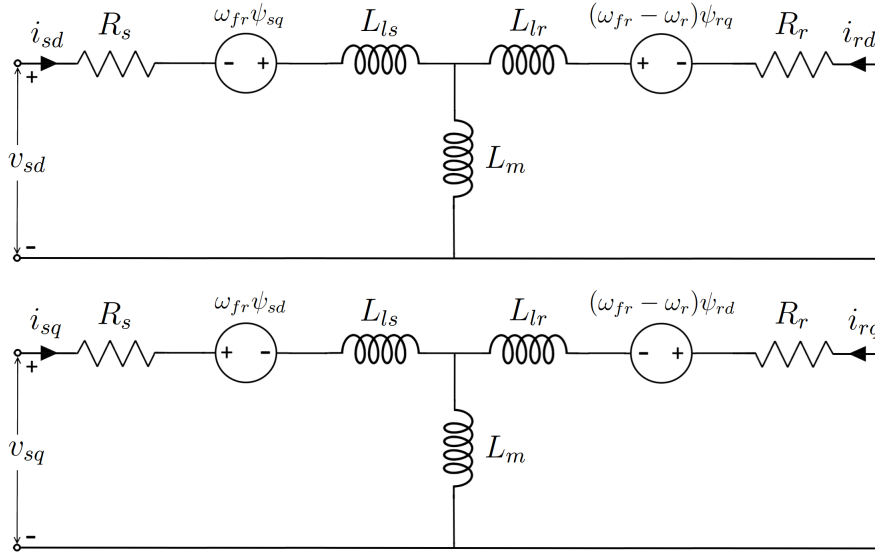
The standard dynamic model of a three-phase induction machine in SI units can be used to describe steady-state as well as transient phenomena. According to [59] the basic equations of a squirrel-cage induction machine can be defined in a coordinate system which rotates with an arbitrary angular velocity  $\omega_{fr}$ . Figure 2.7 illustrates the equivalent circuit of such a machine in the  $dq$  reference frame based on matrix notation.

In matrix notation, the voltage equations of the dynamic model are constituted by

$$\mathbf{v}_s = \begin{bmatrix} v_{sd} \\ v_{sq} \end{bmatrix} = R_s \mathbf{i}_s + \frac{d\boldsymbol{\psi}_s}{dt} + \omega_{fr} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \boldsymbol{\psi}_s \quad (2.56a)$$

$$\mathbf{v}_r = 0 = \begin{bmatrix} v_{rd} \\ v_{rq} \end{bmatrix} = R_r \mathbf{i}_r + \frac{d\boldsymbol{\psi}_r}{dt} + (\omega_{fr} - \omega_r) \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \boldsymbol{\psi}_r \quad (2.56b)$$

where  $\mathbf{v}_s(\mathbf{v}_r)$  denotes the stator (rotor) voltage vector,  $\mathbf{i}_s(\mathbf{i}_r)$  the stator (rotor) current vector,  $\boldsymbol{\psi}_s(\boldsymbol{\psi}_r)$  the stator (rotor) flux vector, and  $R_s(R_r)$  the stator (rotor) winding resistance. Note that for a squirrel-cage induction machine,  $\mathbf{v}_r = 0$  due to the rotor bars

Figure 2.7: Equivalent circuit of an induction machine in the  $dq$  reference frame.

being short-circuited. The stator (rotor) flux linkage equations are described by

$$\boldsymbol{\psi}_s = \begin{bmatrix} \psi_{sd} \\ \psi_{sq} \end{bmatrix} = L_s \mathbf{i}_s + L_m \mathbf{i}_r, \quad (2.57a)$$

$$\boldsymbol{\psi}_r = \begin{bmatrix} \psi_{rd} \\ \psi_{rq} \end{bmatrix} = L_r \mathbf{i}_r + L_m \mathbf{i}_s, \quad (2.57b)$$

with stator and rotor self-inductances

$$L_s = L_{ls} + L_m, \quad (2.58a)$$

$$L_r = L_{lr} + L_m. \quad (2.58b)$$

$L_m$  is the main or magnetising inductance, with  $L_{ls}$  ( $L_{lr}$ ) denoting the stator (rotor) leakage inductance. The electromagnetic machine torque defined in terms of the mechanical load torque  $T_m$  and moment of inertia  $M$ , rotating at an angular velocity equal to the rotor shaft  $\Omega_r$  is given by

$$M \frac{d\Omega_r}{dt} = T_e - T_m. \quad (2.59)$$

Mechanical power is consequently stated as

$$P_m = \Omega_r T_e. \quad (2.60)$$

#### 2.5.4 Per unit definitions

The common practice of normalising all variables and parameters is followed in this work. Normalised variables are chosen as unity when operating a power system at its nominal or rated value.

### 2.5.4.1 Base quantities

The three primary base quantities used in the construction of a per unit (pu) system for a three-phase star connected electrical load are as follows: base voltage  $V_B$  and current  $I_B$  are defined as the peak values of the load's rated phase voltage  $V_R$  and current  $I_R$ . The base angular frequency  $\omega_B$  is set by the rated angular frequency  $\omega_{sR} = 2\pi f_{sR}$ , where  $f_{sR}$  denotes the rated system frequency. Table 2.3 summarises the most commonly used base quantities defined in terms of the noted primaries. The process of normalising can be

Table 2.3: Base values of a pu system in terms of  $V_R$ ,  $I_R$ , and  $\omega_{sR}$ .

Base quantity	Base value
Voltage	$V_B = \sqrt{2/3}V_R$
Current	$I_B = \sqrt{2}I_R$
Angular frequency	$\omega_B = \omega_{sR}$
Resistance, reactance, impedance	$Z_B = V_B/I_B$
Inductance	$L_B = Z_B/\omega_B$
Capacitance	$C_B = 1/\omega_B Z_B$
Apparent power	$S_B = 3/2V_B I_B$
Flux linkage	$\lambda_B = V_B/\omega_B$
Torque	$T_B = \text{pfp}S_B/\omega_B$
Time axis	$t_B = t\omega_B$

confusing at times, especially for an IM. In [5], the necessary steps and sequence to effect an accurate per unit conversion are well described. Noteworthy, the normalisation process maintains the *structure* of the fundamental IM equations, with the exception of the torque equations. Henceforth in this writing pu-values will be implied, unless otherwise stated.

### 2.5.5 Induction machine model

When formulating model predictive control problems, the general approach is to represent the model of an IM in state-space form. To maintain linearity in the state-space equations, the mechanical speed of the IM is assumed to *remain constant* throughout a sample period,

i.e.

$$\frac{d\Omega_r}{dt} = 0. \quad (2.61)$$

As a consequence, the rotor speed  $\Omega_r$  can be seen as a parameter and not a state variable. Representing the model in the orthogonal  $dq$  reference frame rotating with angular frequency  $\omega_{fr}$ , state-variables are selected from the stator and rotor quantities. It is common to select from the stator and rotor currents or flux linkages or a combination thereof to represent the dynamic nature of the IM. Two main approaches are followed in literature. The first opts for selecting the stator flux  $\boldsymbol{\psi}_s$  and rotor flux  $\boldsymbol{\psi}_r$  as state-variables, while the second replaces the stator flux with its current counterpart  $\boldsymbol{i}_s$ . Both dynamic models use a combination of the mentioned variables, of which the inter-relationship is signified by

$$\boldsymbol{i}_s = \frac{1}{D} (L_r \boldsymbol{\psi}_s - L_m \boldsymbol{\psi}_r) \quad (2.62a)$$

$$\boldsymbol{i}_r = \frac{1}{D} (-L_m \boldsymbol{\psi}_s + L_s \boldsymbol{\psi}_r) \quad (2.62b)$$

$$\text{where } D = L_s L_r - L_m^2. \quad (2.62c)$$

From this, the continuous time state-space equations are derived to define the respective IM models.

### 2.5.5.1 Stator flux and rotor flux as state-variables

Dynamic model:

$$\frac{d\boldsymbol{\psi}_s}{dt} = - \left( R_s \frac{L_r}{D} + \omega_{fr} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \right) \boldsymbol{\psi}_s + R_s \frac{L_m}{D} \boldsymbol{\psi}_r + \boldsymbol{v}_s \quad (2.63a)$$

$$\frac{d\boldsymbol{\psi}_r}{dt} = R_r \frac{L_m}{D} \boldsymbol{\psi}_s - \left( R_r \frac{L_s}{D} + (\omega_{fr} - \omega_r) \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \right) \boldsymbol{\psi}_r + \boldsymbol{v}_r. \quad (2.63b)$$

Electromagnetic torque:

$$T_e = \frac{1}{\text{pf}} \frac{L_m}{D} (\boldsymbol{\psi}_r \times \boldsymbol{\psi}_s). \quad (2.64)$$

### 2.5.5.2 Stator current and rotor flux as state-variables

Dynamic model:

$$\frac{d\mathbf{i}_s}{dt} = - \left( \frac{1}{\tau_s} + \omega_{fr} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \right) \mathbf{i}_s + \left( \frac{1}{\tau_r} \mathbf{I}_2 - \omega_r \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \right) \frac{L_m}{D} \boldsymbol{\psi}_r + \frac{L_r}{D} \mathbf{v}_s - \frac{L_m}{D} \mathbf{v}_r \quad (2.65a)$$

$$\frac{d\boldsymbol{\psi}_r}{dt} = \frac{L_m}{\tau_r} \mathbf{i}_s - \left( \frac{1}{\tau_r} + (\omega_{fr} - \omega_r) \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \right) \boldsymbol{\psi}_r + \mathbf{v}_r \quad (2.65b)$$

with time constants of the stator and rotor circuits given by

$$\tau_s = \frac{L_r D}{R_s L_r^2 + R_r L_m^2} \quad \text{and} \quad \tau_r = \frac{L_r}{R_r}. \quad (2.66)$$

Electromagnetic torque:

$$T_e = \frac{1}{\text{pf}} \frac{L_m}{L_r} (\boldsymbol{\psi}_r \times \mathbf{i}_s). \quad (2.67)$$

Note that pu-values are considered and expansion of the cross product is  $\boldsymbol{\psi}_r \times \mathbf{i}_s = (\psi_{rd} i_{sq} - \psi_{rq} i_{sd})$ .

## 2.5.6 Predictive control of an induction machine

In the field of high performance speed control techniques, two methods have been dominating the market in the last decades. One is field oriented control (FOC) [60] and the other direct torque control (DTC) [61]. The most widely used strategy in medium and low power electrical drives is FOC [62]. However, the cascaded structure of FOC inhibits the dynamic response of the control method. One way to reduce the cascaded structure is to use predictive controllers [63].

In the field of electrical drives, two main subcategories of predictive controllers have emerged, namely model predictive current control (MPCC) [64, 65] and model predictive torque control (MPDTC) [66, 67]. Another variant of MPC is model predictive direct current control (MPDCC) [20, 68]. The work presented in this writing will consider the MPCC approach presented in [69] where the mechanical subsystem (outer speed loop) is managed by a PI controller, and the inner current loop by an FCS-MPC based controller. A basic layout of such a system is presented in Figure 2.8.

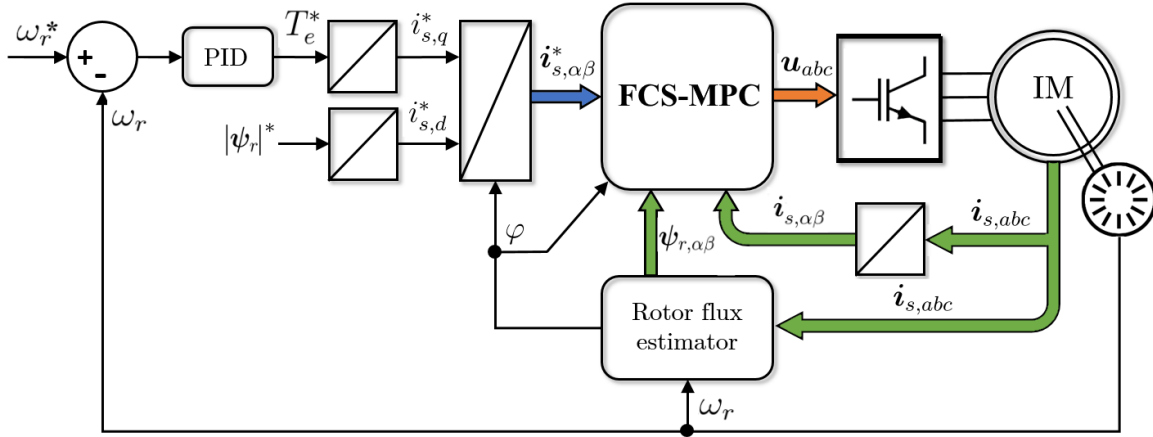


Figure 2.8: Basic Predictive Current Control scheme.

### 2.5.7 Drive performance parameters

Control of an electrical machine can be weighed in terms of the total demand distortion (TDD) of the two metrics listed below.

#### 2.5.7.1 Torque

Controlling the torque of an IM is crucial to tracking the reference speed and maintaining the set speed under load fluctuations. Transient occurrences in particular necessitate a fast response time in adjusting the torque. During steady-state operation, constant torque is required for the purpose of minimising mechanical stress on the motor and load. The varying component of the machine torque is typically measured in terms of the harmonic distortion present in the associated waveform. The measure, when quantified relative to the *rated* machine torque  $T_R$ , is referred to as the total demand distortion

$$T_{TDD} = \frac{1}{T_R} \sqrt{\sum_{n \neq 0}^{\infty} (\hat{T}_{e,n})^2}, \quad (2.68)$$

where  $\hat{T}_{e,n}$  represents the amplitudes of torque harmonics in the frequency spectrum  $nf_1$ ,  $0 < n \in \mathbb{R}$ . Note that  $f_1$  denotes the fundamental frequency of the IM.

#### 2.5.7.2 Stator current

Harmonic distortions in the load current cause losses and give rise to increased temperatures. According to IEEE 519-1992 [70], the total effect of distortion in the current waveform at a point of common connection (PCC) is measured by the total demand distortion as a percentage of the maximum demand current at the PCC. Accordingly, the

the stator current TDD of a *three-phase* IM is defined by

$$I_{TDD} = \frac{1}{3\sqrt{2}I_R} \left( \sqrt{\sum_{n \neq 1}^{\infty} (\hat{i}_{a,n})^2} + \sqrt{\sum_{n \neq 1}^{\infty} (\hat{i}_{b,n})^2} + \sqrt{\sum_{n \neq 1}^{\infty} (\hat{i}_{c,n})^2} \right), \quad (2.69)$$

where  $I_R$  denotes the rated stator current, and  $\hat{i}_{x,n}$  the harmonic amplitude of the individual stator currents  $x = a, b, c$  in the frequency spectrum  $nf_1$ ,  $n \neq 1$ . Again, the fundamental  $f_1$ , denotes the fundamental frequency.

### 2.5.8 Chapter summary

This chapter introduced a generic approach to the model predictive control of power electronic systems. In particular, the underlying optimisation problem associated with long-horizon FCS-MPC was highlighted and presented as an integer least-squares problem. The chapter concluded with the description and modeling of the physical systems related to this work.



# Chapter 3

## Search space and solvers

Solving the optimisation problem underlying FCS-MPC (2.36) equates to solving an ILS problem. In lattice theory, the ILS problem is one of the fundamental problems which has been investigated intensively, and is commonly referred to as the *closest vector problem* (CVP) [71]. The following sections introduce lattice fundamentals, their structure and space partitioning as prologue to the task of solving the CVP. Consideration of various solvers to this problem follows with the presentation of a novel projection algorithm to conclude the chapter.

### 3.1 Lattice fundamentals

An  $n$ -dimensional lattice  $\mathcal{L}$  is a discrete set of points with a periodic structure, located in an  $m$ -dimensional space ( $m \geq n$ ) [72]. The structure of a lattice is defined by a set of linearly independent vectors  $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n$  in  $\mathbb{R}^m$  which is referred to as the *basis of a lattice*. A lattice exhibits a periodic nature which stems from integer combinations of the basis vectors as defined by the general definition

$$\mathcal{L}(\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n) = \left\{ \sum_{j=1}^n \tilde{u}_j \mathbf{b}_j \mid \tilde{u}_j \in \mathbb{Z} \right\}. \quad (3.1)$$

Indexing the basis vectors as columns in an  $\mathbb{R}^{m \times n}$  matrix  $\mathbf{B} = [\mathbf{b}_1 \ \mathbf{b}_2 \ \dots \ \mathbf{b}_n]$ , allows for the simplified definition

$$\mathcal{L}(\mathbf{B}) = \{ \mathbf{B}\tilde{\mathbf{u}} \mid \tilde{\mathbf{u}} \in \mathbb{Z}^n \}, \quad (3.2)$$

where  $\mathbf{B}$  is referred to as the *lattice generator matrix*. Due to the unique nature of the generator matrix  $\mathbf{H}$  in this work (2.34b), only square invertible matrices will be

considered, i.e. matrices that generate *full-dimensional* lattices. Hence, we reintroduce the symbol  $d$  for denoting the dimension of both, the lattice ( $n$ ) and the space ( $m$ ). As an example, consider the simplest non-trivial hyper-cubic lattice  $\mathbb{Z}^d$ , which is formally defined as

$$\mathcal{L}(\mathbf{I}_d) = \{\mathbf{I}_d \tilde{\mathbf{u}} \mid \tilde{\mathbf{u}} \in \mathbb{Z}^d\} \quad (3.3)$$

with the  $d \times d$  identity matrix  $\mathbf{I}_d = [\mathbf{e}_1 \ \mathbf{e}_2 \ \dots \ \mathbf{e}_d]$  as generator matrix. The column unit vectors of  $\mathbf{I}_d$  form the lattice basis which is also the natural basis for Euclidean space  $\mathbb{R}^d$ . The resulting lattice is one that consists of all points in  $\mathbb{R}^d$  with *integer* coordinates. Fig. 3.1a exemplifies the two-dimensional case ( $d = 2$ ), with the basis vectors i.e columns of the generator matrix  $\mathbf{I}_d$  indicated as blue arrows.

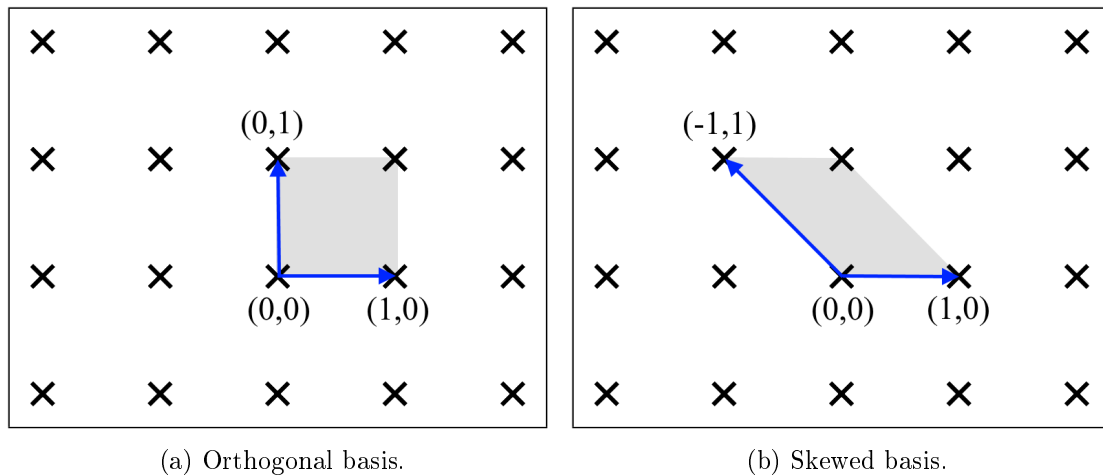


Figure 3.1: Square lattice  $\mathbb{Z}^2$  with two different sets of basis vectors.

In Figure 3.1b the same lattice is displayed, but with a different set of basis vectors. The shaded areas in both figures are referred to as the fundamental or basic *paralleloptope*<sup>5</sup> of the relevant lattice basis and is defined by

$$\mathcal{P}(\mathbf{B}) = \{\mathbf{B}\mathbf{x} \mid \mathbf{x} \in \mathbb{R}^d \text{ for all } 0 \leq x_j < 1\}. \quad (3.4)$$

As any lattice of dimension two or more has infinitely many bases [74, p.205], definition (3.4) substantiates a simple lemma for determining if a set of linearly independent vectors defines a valid basis. According to Micciancio and Goldwasser(2012) [75, p.4] no other

<sup>5</sup>A paralleloptope is considered as the generalisation of a parallelogram in higher dimensions [73]. In  $d$ -dimensional space it is called a  $d$ -paralleloptope. A parallelogram is therefore a 2-paralleloptope and a parallelepiped is a 3-paralleloptope.

lattice point, except the origin, is to be contained in the basic parallelotope generated by the basis vectors. It also follows that the vector space spanned by the lattice

$$\text{span}(\mathcal{L}(\mathbf{B})) = \{\mathbf{B}\mathbf{x} \mid \mathbf{x} \in \mathbb{R}^d\}, \quad (3.5)$$

can be tiled with shifted copies of  $\mathcal{P}(\mathbf{B})$ , one per lattice vector [76, p.11]. Often, a lattice is *limited* or *truncated* to a subset  $\mathbb{U}$  of the infinite integer lattice  $\mathbb{Z}^d$ . Chosen to coincide with physical systems presented in this work, consider the integer set

$$\mathcal{U} = \{1, 0, -1\}, \quad (3.6)$$

defining the subset

$$\mathbb{U} = \mathcal{U}^d \subset \mathbb{Z}^d, \quad (3.7)$$

where  $\mathcal{U}^d$  denotes the set  $\mathcal{U}$  to the Cartesian power  $d$ , i.e.  $\mathcal{U}^d = \mathcal{U}_1 \times \dots \times \mathcal{U}_d$ . With cardinality of the set  $\mathcal{U}$  equaling three ( $\#\mathcal{U} = 3$ ), the set  $\mathbb{U}$  holds  $\#\mathbb{U} = 3^d$  integer combinations. Limiting the hyper-cubic lattice (3.3), with the subset  $\mathbb{U}$  results in a truncated lattice

$$\mathcal{L}'(\mathbf{I}_d) = \{\mathbf{I}_d\tilde{\mathbf{u}} \mid \tilde{\mathbf{u}} \in \mathbb{U}\}. \quad (3.8)$$

The *convex hull* of a finite set of lattice points is defined as the smallest *convex polytope*<sup>6</sup> that contains all points of the set. Alternatively, Krein(1940) [77] states that every convex set in a Euclidean space is the convex hull of its extreme points. The convex hull of the lattice  $\mathcal{L}'(\mathbf{I}_d)$  can therefore be represented as

$$\mathcal{P}(\mathcal{L}'(\mathbf{I}_d)) = \{\mathbf{I}_d\mathbf{x} \mid \mathbf{x} \in \mathbb{R}^d \text{ for all } \|\mathbf{x}\|_\infty \leq \mathcal{U}_{max}\}, \quad (3.9)$$

with its extreme points or outer bound defined by

$$\mathcal{P}'(\mathcal{L}'(\mathbf{I}_d)) = \{\mathbf{I}_d\mathbf{x} \mid \mathbf{x} \in \mathbb{R}^d \text{ for all } \|\mathbf{x}\|_\infty = \mathcal{U}_{max}\}. \quad (3.10)$$

Note that  $\mathcal{U}_{max}$  identifies the element of maximum value in the set  $\mathcal{U}$  (3.6). Resembling a  $d$ -dimensional box or hypercube, Figure 3.2 depicts the two-dimensional case.

<sup>6</sup>A polytope is defined by [73] as the general term of the sequence “point, line segment, polygon, polyhedron,...”. More specifically, a *convex polytope* is defined as a finite region of  $d$ -dimensional space enclosed by a finite number of  $(d - 1)$ -dimensional hyperplanes.

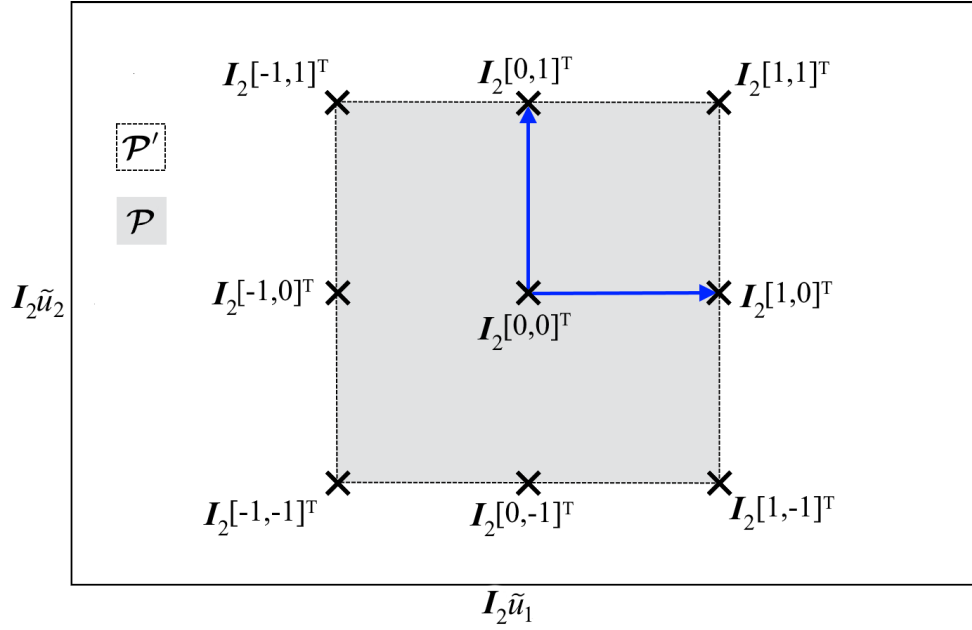


Figure 3.2: *Truncated lattice* with orthonormal basis  $\mathbf{I}_d$  in  $\mathbb{R}^2$ .

## 3.2 Lattice structure

Hassibi(2005) [78] states that, the transformation induced by any lattice-generating matrix may result in vectors that span a *skewed* lattice. In [72, p.360], skewness of a lattice is quantified in terms of the *orthogonality defect* (OD) of its basis, i.e.

$$\sigma(\mathbf{B}) = \prod_{j=1}^d \|\mathbf{b}_j\| / \det(\mathbf{B}) \geq 1, \quad (3.11)$$

with equality if the lattice basis vectors  $\mathbf{b}_j$  are orthogonal to one another. The lattice  $\mathcal{L}'(\mathbf{I}_d)$  with orthonormal basis  $\mathbf{I}_d$  will therefore pose an orthogonality defect of  $\sigma(\mathbf{I}_d) = 1$ . Alternatively, as an example, consider some  $\mathbb{R}^{d \times d}$  matrix  $\mathbf{H} = [\mathbf{h}_1 \ \mathbf{h}_2 \ \dots \ \mathbf{h}_d]$  with an orthogonality defect  $\sigma(\mathbf{H}) > 1$ , as generator to the lattice

$$\mathcal{L}(\mathbf{H}) = \{\mathbf{H}\tilde{\mathbf{u}} \mid \tilde{\mathbf{u}} \in \mathbb{U}\}. \quad (3.12)$$

Figure 3.3 illustrates a two-dimensional variant of such a skewed, truncated lattice. The lattice basis vectors are indicated in blue, and the resulting *convex hull* is highlighted by the shaded area. The convex hull takes the shape of a  $d$ -dimensional parallelotope, which is bound by a set of  $(2 \times d)$ ,  $(d - 1)$ -dimensional affine hyperplanes. Due to the structure of the truncated lattice  $\mathcal{L}(\mathbf{H})$ , these bounding hyperplanes are translations of the transformed coordinate system's axis-hyperplanes

$$\mathcal{A}_j = \{\mathbf{x} \in \mathbb{R}^d : \langle \mathbf{n}_j, \mathbf{x} \rangle = 0\} \quad \text{with } j = 1, 2, \dots, d. \quad (3.13)$$

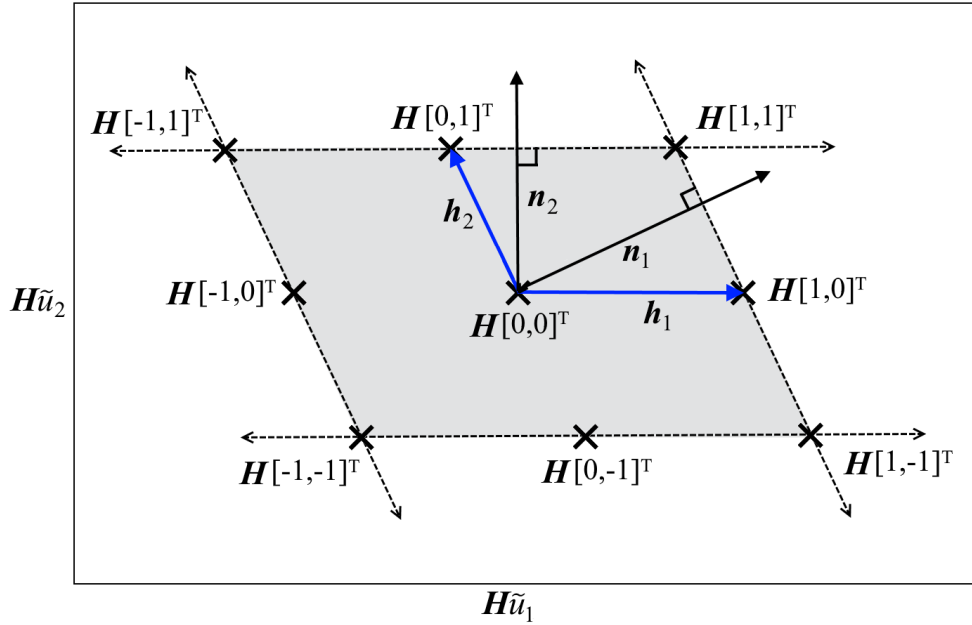


Figure 3.3: A truncated lattice with skew basis in  $\mathbb{R}^2$ . Bounding the convex hull is a set of  $(2 \times d)$ ,  $(d - 1)$ -dimensional affine hyperplanes.

Note that  $\mathbf{n}_j$  denotes the normal vectors to the respective planes, and that the statement  $\langle \mathbf{n}_j \cdot \mathbf{x} \rangle = 0$ , geometrically speaking, translates to the orthogonal projection of the vector  $\mathbf{x}$  onto the normal  $\mathbf{n}_j$ . A result equaling zero implies that the corresponding axis-hyperplane resides in the *null-space* of  $\mathbf{n}_j$ . For more information on orthogonal projection in linear spaces the reader is referred to Appendix A. The normal vectors  $\mathbf{n}_j$  also represent translations of their respective axis-hyperplanes. Hence, the affine hyperplanes bounding the convex hull can be presented as

$$\mathcal{H}_j = \{ \mathbf{x} \in \mathbb{R}^d : \langle \mathbf{n}_j \cdot (\mathbf{x} + \mathbf{h}_j) \rangle = 0 \} \quad \text{and} \quad (3.14a)$$

$$-\mathcal{H}_j = \{ \mathbf{x} \in \mathbb{R}^d : \langle \mathbf{n}_j \cdot (\mathbf{x} - \mathbf{h}_j) \rangle = 0 \} \quad \text{with } j = 1, 2, \dots, d. \quad (3.14b)$$

Note that the respective affine hyperplanes are defined in *point-normal* form with  $\pm \mathbf{h}_j$  referencing the translations of the axis-hyperplanes. Henceforth, the set of  $\pm \mathcal{H}_j$  will be referred to as the *hull-hyperplanes* of the lattice convex hull. Due to the symmetry of the convex hull and for the purpose of mathematical manipulation, the normals to the hull-hyperplanes are presented as a set of vectors indexed as the columns of the matrix

$$\mathbf{N} = [\mathbf{n}_1 \ \mathbf{n}_2 \ \dots \ \mathbf{n}_d]. \quad (3.15)$$

Obtaining the normals in the transformed space calls on the transformation rules applicable to perpendicular planes. This is because *normal vectors* do not transform in the same

way as points or vectors do [79]. Moreover, a separate *normal transformation matrix* is needed, which as a standard is defined as the *transposed inverse* of the generator matrix. Appendix A.4 elaborates on this standard transformation. With reference to the cubic lattice defined in (3.8), we recall that the basis vectors in the generator matrix  $\mathbf{I}_d$  are orthogonal and  $\sigma(\mathbf{I}_d) = 1$ . Significance of an orthogonal basis lies therein that the respective basis vectors are perpendicular to the axis hyperplanes, and therefore also double as normal vectors, i.e.

$$\mathbf{N}_I = \mathbf{I}_d^{-T} = \mathbf{I}_d. \quad (3.16)$$

Employing a lattice generator matrix  $\mathbf{H}$  with an orthogonality defect exceeding unity  $\sigma(\mathbf{H}) > 1$ , will result in a transformed coordinate space where the basis vectors are *not perpendicular* to the axis-hyperplanes, and therefore differ from the transformed normals

$$\mathbf{N} = \mathbf{H}^{-T} \mathbf{N}_I = \mathbf{H}^{-T}. \quad (3.17)$$

Transformation of the normals from the natural basis to the transformed basis, rotates the normals to maintain orthogonality to the respective axis-hyperplanes. In the transformation process, the norms of the normal vectors are not preserved due to scaling imposed by the matrix inverse. In general, this is of no consequence, as a normal to a hyperplane is required to be nonzero, and to convey directional information. However, for the purpose of defining the convex hull in terms of the normal vectors, it is necessary to consider the scaling that occurred. Projecting the basis vectors of  $\mathbf{H}$  orthogonally onto the corresponding normal vectors  $\mathbf{N}$  results in the normal vector components

$$\mathbf{n}_{j,h} = \frac{\langle \mathbf{n}_j \cdot \mathbf{h}_j \rangle}{\langle \mathbf{n}_j \cdot \mathbf{n}_j \rangle} \mathbf{n}_j. \quad (3.18)$$

From 3.17 note that

$$\mathbf{N}^T \mathbf{H} = (\mathbf{H}^{-T})^T \mathbf{H} = \mathbf{I}, \quad (3.19)$$

which signifies

$$\langle \mathbf{n}_j \cdot \mathbf{h}_j \rangle = 1, \quad \text{for } j = 1, 2, \dots, d. \quad (3.20)$$

Updating (3.18) leave

$$\mathbf{n}_{j,h} = \frac{1}{\langle \mathbf{n}_j \cdot \mathbf{n}_j \rangle} \mathbf{n}_j = k_{j,h} \mathbf{n}_j, \quad (3.21)$$

where the *projection coefficient*

$$k_{j,h} = \frac{1}{\|\mathbf{n}_j\|^2}, \quad (3.22)$$

represents the scaling of  $\mathbf{n}_j$  as such that  $k_{j,h} \|\mathbf{n}_j\|$  conveys the distance of the hull hyperplane, defined by  $\mathbf{n}_j$ , from the origin. Also consider the orthogonal projection of  $\mathbf{x} \in \mathbb{R}^d$  onto the set of normal vectors  $\mathbf{N}$  to give

$$\mathbf{n}_{j,x} = \frac{\langle \mathbf{n}_j \cdot \mathbf{x} \rangle}{\langle \mathbf{n}_j \cdot \mathbf{n}_j \rangle} \mathbf{n}_j = k_{j,x} \mathbf{n}_j, \quad (3.23)$$

with the *projection coefficient*

$$k_{j,x} = \frac{\langle \mathbf{n}_j \cdot \mathbf{x} \rangle}{\|\mathbf{n}_j\|^2}. \quad (3.24)$$

From (3.21) and (3.23) it can be concluded that if  $\mathbf{x}$  co-exists with  $\mathbf{h}_j$  in the hull hyperplane defined by  $\mathbf{n}_j$ , then

$$\mathbf{n}_{j,x} = \mathbf{n}_{j,h} \quad (3.25a)$$

$$k_{j,x} = k_{j,h} \quad (3.25b)$$

$$\langle \mathbf{n}_j \cdot \mathbf{x} \rangle = 1. \quad (3.25c)$$

Utilising this simplification, the set of hull hyperplanes (3.14) can be restated as

$$\mathcal{H}_j(\mathcal{L}(\mathbf{H})) = \{\mathbf{x} \in \mathbb{R}^d : \langle \mathbf{n}_j \cdot \mathbf{x} \rangle = +1\} \quad \text{and} \quad (3.26a)$$

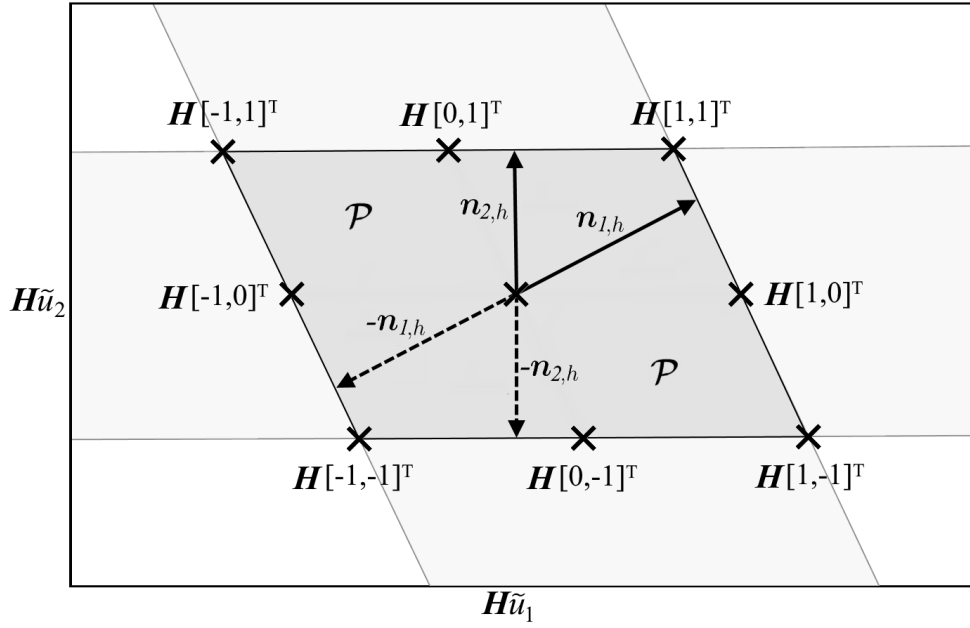
$$-\mathcal{H}_j(\mathcal{L}(\mathbf{H})) = \{\mathbf{x} \in \mathbb{R}^d : \langle \mathbf{n}_j \cdot \mathbf{x} \rangle = -1\} \quad \text{with } j = 1, 2, \dots, d. \quad (3.26b)$$

The combined set  $\pm\mathcal{H}$  *supports*<sup>7</sup> the convex hull in an inward direction, hence the characterisation of the convex hull

$$\mathcal{P}(\mathcal{L}(\mathbf{H})) = \{\mathbf{x} \in \mathbb{R}^d : |\langle \mathbf{n}_j \cdot \mathbf{x} \rangle| \leq 1, \quad \text{for } j = 1, 2, \dots, d\}. \quad (3.27)$$

Geometrically (3.27) translates into the intersection of *inner* half-spaces which are supported by the positive and negative hyperplane pairs  $\pm\mathcal{H}_j$ , located at a distance of  $k_{j,h} \|\mathbf{n}_j\|$  from the origin. Note that the intersection of a pair of inner half-spaces constitutes a *slice* through  $\mathbb{R}^d$  space and that the combined intersection of all slices results in the convex hull of the lattice. Figure 3.4 demonstrates the geometrical interpretation for the two-dimensional case. With further consideration of (3.27) and (3.15), we introduce

<sup>7</sup>The hyperplane  $\mathcal{H}$  or boundary of a half-space that contains an arbitrary convex set  $\mathcal{P}$  is called a *supporting hyperplane* to  $\mathcal{P}$  when the hyperplane contains at least one point of  $\mathcal{P}$  [80].

Figure 3.4: Lattice span as intersection of half-spaces in  $\mathbb{R}^2$ .

the vector

$$\mathbf{k} = \mathbf{N}^T \mathbf{x}, \quad (3.28)$$

of which the elements  $k_j = \langle \mathbf{n}_j \cdot \mathbf{x} \rangle$ , convey the dimensional information of  $\mathbf{x}$  as a component of  $\mathbf{n}_j$ . This gives way for the formulation of a simple *containment test*

$$\|\mathbf{k}\|_\infty \leq 1 \quad (3.29)$$

which verifies the residency of  $\mathbf{x} \in \mathcal{P}$ . If  $\|\mathbf{k}\|_\infty \leq 1$  then  $\mathbf{x}$  is contained by the convex hull and if  $\|\mathbf{k}\|_\infty > 1$  then  $\mathbf{x}$  is obviously outside. If  $\mathbf{x}$  resides in the outer bound of the convex hull, then  $\|\mathbf{k}\|_\infty = 1$ . Significance of this test will become clear in the projection algorithm presented later in this chapter.

### 3.3 Closest vector problem

The closest vector problem which is a classic mathematical problem in the study of the geometry of numbers [81] refers to searching in a lattice, i.e. the *lattice search space*, for the lattice vector with minimum Euclidean distance to a given target. The problem of interest to this study is finding for the lattice

$$\mathcal{L}(\mathbf{H}) = \{ \mathbf{H}\tilde{\mathbf{u}} \mid \tilde{\mathbf{u}} \in \mathbb{Z}^d \}, \quad (3.30)$$



and a given target vector  $\mathbf{x} \in \mathbb{R}^d$ , a lattice point  $\hat{\mathbf{s}} \in \mathcal{L}(\mathbf{H})$  in an Euclidean sense that is closest to the target vector, i.e.

$$\|\hat{\mathbf{s}} - \mathbf{x}\| \leq \|\mathbf{s} - \mathbf{x}\| \quad \text{for all } \mathbf{s} \in \mathcal{L}(\mathbf{H}). \quad (3.31)$$

Equivalently, the CVP can be formulated as such that the *squared Euclidean norm* is minimised to give the closest lattice point

$$\hat{\mathbf{s}} = \arg \min_{\mathbf{s} \in \mathcal{L}(\mathbf{H})} \|\mathbf{s} - \mathbf{x}\|^2. \quad (3.32)$$

An extension of this formulation which joins onto the constrained ILS problem stated in (2.36a), is finding the optimal integer combination  $\hat{\mathbf{u}}$  that describes the lattice point  $\hat{\mathbf{s}} = \mathbf{H}\hat{\mathbf{u}}$ , with minimum distance to the target vector  $\mathbf{x} = \mathbf{H}\mathbf{u}_{unc}$ , i.e.

$$\hat{\mathbf{u}} = \arg \min_{\tilde{\mathbf{u}} \in \mathbb{U}} \|\mathbf{H}\tilde{\mathbf{u}} - \mathbf{x}\|^2. \quad (3.33)$$

### 3.4 Space partitioning

Solving the general closest vector problem in a lattice requires exploration of the local neighbourhood in a geometric space. Aurenhammer (1991) [82] argues that this can be done efficiently with a *Voronoi diagram*. A Voronoi diagram starts out with a given set of points, also referred to as *sites* or *seeds* in Euclidean space. This space is uniquely partitioned into disjoint polytopes to such an extent that each polytope is assigned to one single site and covers all the points in the space that are closer to that specific site than to any other site. Each of the resulting polytopes is called a *Voronoi or Dirichlet cell*. The Voronoi cell  $\mathcal{V}$  of an individual site  $\hat{\mathbf{s}} \in \mathcal{L}$ , can be defined as [83]

$$\mathcal{V}(\mathcal{L}, \hat{\mathbf{s}}) = \{\mathbf{x} \in \mathbb{R}^d : \|\hat{\mathbf{s}} - \mathbf{x}\| \leq \|\mathbf{s} - \mathbf{x}\| \quad \text{for all } \mathbf{s} \in \mathcal{L}, \mathbf{s} \neq \hat{\mathbf{s}}\}. \quad (3.34)$$

The Voronoi diagram for the two-dimensional truncated lattice  $\mathcal{L}(\mathbf{H})$  of (3.12) is presented in Figure 3.5. Note that each lattice point  $\mathbf{s} = \mathbf{H}\tilde{\mathbf{u}}$  is a Voronoi site, and that the Voronoi cell of the origin  $\mathbf{H}\tilde{\mathbf{u}} = \mathbf{0}$  is the only bounded cell. Due to the lattice  $\mathcal{L}(\mathbf{H})$  being truncated with  $\tilde{\mathbf{u}} \in \mathbb{U}$ , all the other Voronoi cells are unbounded, as their respective sites constitute the outer layer of the lattice.

The Voronoi cell enclosing the origin is also known as the *basic Voronoi cell* of a lattice, which is a convex polytope, symmetrical in reflection through its Voronoi site, i.e.

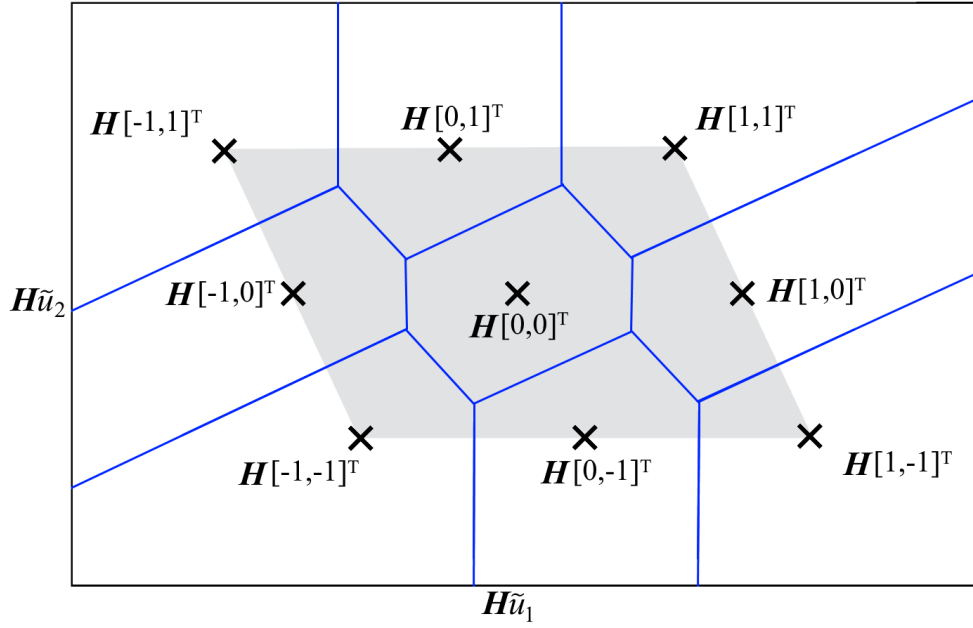


Figure 3.5: Voronoi diagram for the set of  $HU$ -sites in  $\mathbb{R}^2$  space.

the origin  $\underline{\mathbf{0}}$  [84, 85, 86]. Substituting  $\hat{\mathbf{s}} = \underline{\mathbf{0}}$  in (3.34) defines the basic Voronoi cell of a lattice

$$\mathcal{V}(\mathcal{L}, \underline{\mathbf{0}}) = \{ \mathbf{x} \in \mathbb{R}^d : \|\underline{\mathbf{0}} - \mathbf{x}\| \leq \|\mathbf{s} - \mathbf{x}\| \quad \text{for all } \mathbf{s} \in \mathcal{L}, \mathbf{s} \neq \underline{\mathbf{0}} \}. \quad (3.35)$$

*Voronoi-relevant vectors*  $\mathbf{v}$  are lattice points closest to a specific lattice point, i.e. lattice points that are adjacent or neighbours to the specific lattice point. A set of Voronoi-relevant vectors is defined as a minimal set  $\mathbf{V}(\mathcal{L})$  for which [86]

$$\mathcal{V}(\mathcal{L}, \underline{\mathbf{0}}) = \{ \mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\| \leq \|\mathbf{x} - \mathbf{s}\| \quad \text{for all } \mathbf{s} \in \mathbf{V}(\mathcal{L}) \}. \quad (3.36)$$

In [87, p.88-96] it was shown that a maximum of  $(2^{d+1} - 2)$  Voronoi-relevant vectors  $\mathbf{v} \in \mathbf{V}$  can exist in a  $d$ -dimensional lattice. Each Voronoi-relevant vector is bisected orthogonally at its midpoint by an affine hyperplane that defines the border between the Voronoi site and its Voronoi-relevant vector. For the two-dimensional example of  $\mathcal{L}(\mathbf{H})$ , Figure 3.6 displays the six Voronoi-relevant vectors to the origin. The vectors are indicated by the blue arrows with the bisecting hyperplanes defining the borders or facets of the basic Voronoi cell. Translation of the basic Voronoi cell to all other lattice points is illustrated by the dotted lines. This notion is described in [88] as lattice tiling with a *prototile*. The Voronoi-relevant vectors are lattice points and thus constitute a subset of the relevant lattice. Stating the basic Voronoi cell in terms of the Voronoi-relevant vectors gives [86]

$$\mathcal{V}(\mathcal{L}, \underline{\mathbf{0}}) = \{ \mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\| \leq \|\mathbf{x} - \mathbf{v}\| \quad \text{for all } \mathbf{v} \in \mathbf{V} \}. \quad (3.37)$$

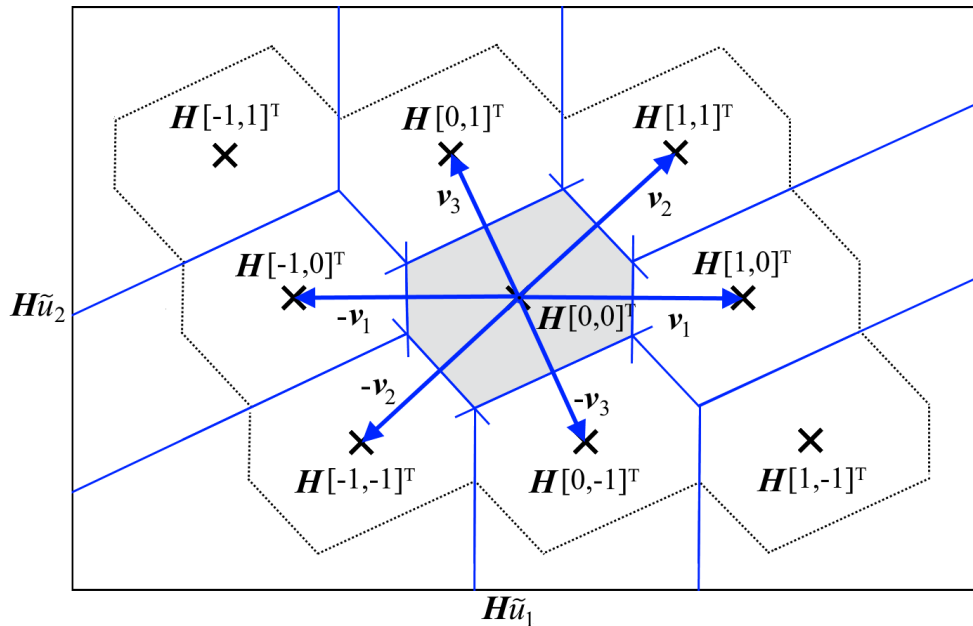


Figure 3.6: Voronoi diagram of a lattice with basic Voronoi cell and relevant vectors.

Due to the symmetrical nature of the Voronoi cell, the Voronoi-relevant vectors are reflections, and for every  $\mathbf{v}$ , the vector  $-\mathbf{v}$  is also a Voronoi-relevant vector. This allows for minimising the set  $\mathbf{V}$  by using only one representative of each Voronoi-relevant *vector pair*  $\{\mathbf{v}, -\mathbf{v}\}$ . Subsequently, the set  $\mathbf{V}$  is updated to represent the set of Voronoi-relevant vector pairs, and now consists of  $m = (2^d - 1)$  *single-sided* Voronoi-relevant vectors. These vectors are indexed as the column vectors of the matrix

$$\mathbf{V} = [\mathbf{v}_1 \ \mathbf{v}_2 \ \dots \ \mathbf{v}_m], \quad m = (2^d - 1). \quad (3.38)$$

With the basic Voronoi cell's borders bisecting the Voronoi-relevant vectors at  $\frac{1}{2}\mathbf{v}$ , the Voronoi cell can be stated in terms of inner half-spaces [86]

$$\mathcal{V}(\mathcal{L}, \underline{\mathbf{0}}) = \left\{ \mathbf{x} \in \mathbb{R}^d : \frac{|\langle \mathbf{v}_j, \mathbf{x} \rangle|}{\|\mathbf{v}_j\|^2} \leq \frac{1}{2} \text{ for all } \mathbf{v}_j, j = 1, 2, \dots, m \right\}. \quad (3.39)$$

The significance of the Voronoi cell lies in its symmetry and translation throughout the relevant lattice. These properties make it possible to determine the entire geometric structure of a lattice from the description of its Voronoi cell [86]. Following the same approach as in (3.29), the definitive term in (3.39) can be considered as a vector of projection coefficients

$$\mathbf{k} = [k_1 \ k_2 \ \dots \ k_m]^T \quad (3.40a)$$

$$\text{where } k_j = \frac{\langle \mathbf{v}_j, \mathbf{x} \rangle}{\|\mathbf{v}_j\|^2} \text{ for } j = 1, 2, \dots, m \quad (3.40b)$$

which can be employed as a *containment test*

$$\|\mathbf{k}\|_{\infty} \leq \frac{1}{2} \quad (3.41)$$

for residency of  $\mathbf{x} \in \mathcal{V}(\mathcal{L}, \mathbf{0})$ .

## 3.5 Decoding strategies

Presentation of the transformed search space in the previous section, and partitioning it into local neighbourhoods or Voronoi cells with a maximum of  $(2^{d+1} - 2)$  borders per cell, hint at the complexity of the CVP into higher dimensions. In literature the CVP has been shown to be NP-hard in obtaining an exact solution [89, 90], or a close approximation thereof [91, 92]. Solving this *hard* problem requires an efficient solver, especially in real-time applications with short sampling times. In the following sub-sections different solvers are considered, in particular *polynomial-space* and *exponential-space* algorithms. For all the solvers presented, the objective will be to solve the truncated CVP

$$\hat{\mathbf{u}} = \arg \min_{\tilde{\mathbf{u}} \in \mathbb{U}} \|\mathbf{H}\tilde{\mathbf{u}} - \mathbf{x}\|^2 \quad (3.42a)$$

$$\text{subject to } \mathbb{U} = \mathcal{U}^d \subset \mathbb{Z}^d, \quad (3.42b)$$

$$\mathcal{U} = \{-1, 0, 1\}. \quad (3.42c)$$

From (3.33), we recall that  $\mathbf{H}\tilde{\mathbf{u}}$  defines the lattice search space, whilst  $\mathbf{x} = \mathbf{H}\mathbf{u}_{unc}$  denotes the CVP target.

### 3.5.1 Sphere decoding algorithm

Representing polynomial-space algorithms, the Sphere Decoding algorithm is considered in this work due to its proven reputation as a practical tool for solving the closest vector problem. It is widely used in communication applications, and has recently been adapted by [39] as a solver to the integer least squares problem specific to the FCS-MPC problem in power electronic systems. In [93, 94] it was considered as solver in a long-horizon FCS-MPC system with a passive *RL*-load, and, more recently, another more complex implementation called on the sphere decoding algorithm to assist in a long-horizon FCS-MPC, induction machine drive [95].

### 3.5.1.1 Concept

The operating principle of an SDA is relatively simple: given a target vector  $\mathbf{x}$  in a  $\mathbf{H}$ -coordinate solution space, the SDA examines only lattice points that reside inside a sphere with *selected radius*  $\delta$  centered around the target. This effectively reduces the search space, and hence the computations required to solve the CVP. Figure 3.7 illustrates the principle for the two-dimensional ( $d = 2$ ) example with the lattice structure truncated in each dimension by the set  $\mathcal{U}$ .

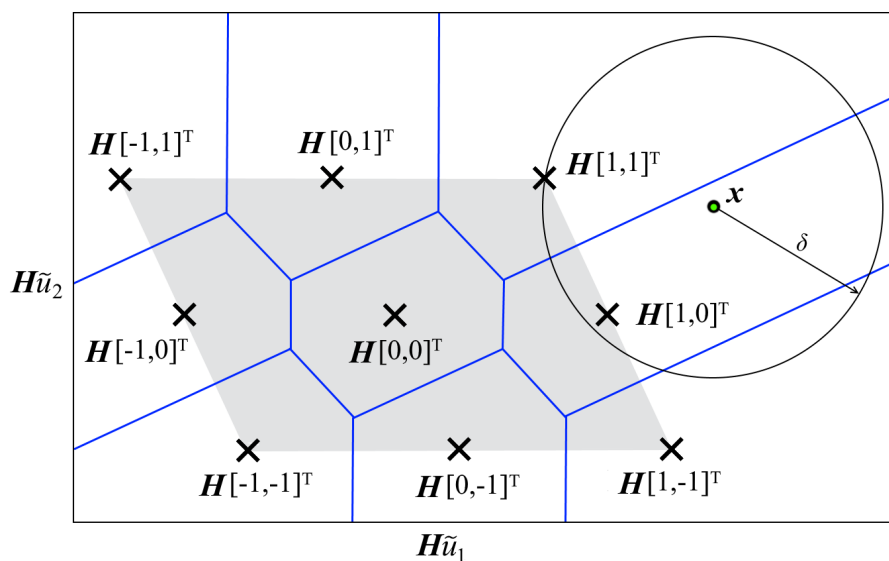


Figure 3.7: Lattice points  $\mathbf{H}[1\ 1]^T$  and  $\mathbf{H}[1\ 0]^T$  enclosed by a sphere with radius  $\delta$  from the target  $\mathbf{x}$ .

Determining which lattice points reside inside a  $d$ -dimensional sphere is a daunting task, but it is trivial to do so in one dimension. The SDA exploits this principle whereby all lattice points in a  $d$ -dimensional sphere and radius  $\delta$  can be found by successively obtaining all the lattice points residing in the spheres of lower dimensions  $n = 1, \dots, d$ , with the same radius. Essentially, the SDA constructs a search tree where the nodes in the  $n^{\text{th}}$  level of the tree correspond to the lattice points inside the sphere of radius  $\delta$  and dimension  $n$ . Figure 3.8 shows a search tree for the *two-dimensional* example, with the highlighted tree branches indicating the paths to the respective nodes which correspond to the lattice points that are found inside the sphere, as demonstrated in Figure 3.7. These lattice points or *leaf-nodes* in the lowest tree level ( $n = d$ ) are considered as candidate solutions.

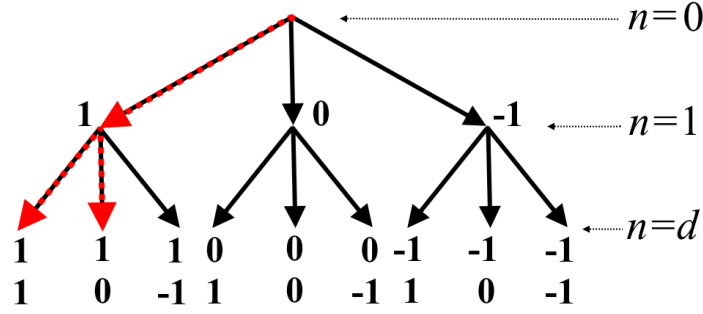


Figure 3.8: Search tree for a two-dimensional ( $d = 2$ ) CVP with limited search space.

The Sphere Decoding algorithm proposed in [39] searches for an optimal solution by considering traversal from one tree level to the next. This is done in a top-down manner or so-called depth-first traversal. Starting at the root node, the algorithm traverses through the tree with a goal to reach the bottom level of the tree (a leaf node) while accumulating the least possible cost. The algorithm in [39] exploits the triangular nature of the generator matrix  $\mathbf{H}$ , and computes the cost of the incumbent solution in the form of the squared distance from the target  $\mathbf{x}$  by

$$J_d^2 = \underbrace{(h_{(1,1)}\tilde{u}_1 - x_1)^2}_{J_1^2} + \underbrace{(h_{(2,1)}\tilde{u}_1 + h_{(2,2)}\tilde{u}_2 - x_2)^2}_{J_2^2} + \dots \quad (3.43)$$

$$+ (h_{(d,1)}\tilde{u}_1 + h_{(d,2)}\tilde{u}_2 + \dots + h_{(d,d)}\tilde{u}_d - x_d)^2.$$

Note that the cost is computed per tree level  $J_n^2$ , or single dimension of the search space. The algorithm economises by accumulating the costs incurred for traversing to previous tree levels, and continuously measures the total cost against the square of the selected sphere radius. As long as the path traversed to a specific node results in a cost  $J_n^2 < \delta^2$ , the algorithm continues to visit nodes at the lower tree levels of the current branch. If the cost is more, the algorithm prunes the branch and backtracks to a higher-level node in an attempt to find a new path to the leaf nodes. Once a path from the root-node to a leaf-node is discovered with a cost  $J_d^2 < \delta^2$ , it is considered as a candidate solution. The sphere radius is updated with the relevant cost of traversal and tightens the sphere to effect more aggressive pruning of the tree. The algorithm continues to search for different paths that may result in lesser cost until the tree is pruned to an extent where only one path through it remains, i.e. to a leaf node that identifies the optimal solution  $\hat{\mathbf{u}}$ .

A crucial aspect to the effectiveness of the SDA is the choice of an initial sphere radius.

This important selection will be discussed in the sub-section on computational complexity which follows after the algorithm description.

### 3.5.1.2 Algorithm

The Sphere Decoding algorithm proposed in [39] is listed as Algorithm 1. It is initiated with the argument  $\text{SPHDEC}(\[], 0, 1, \delta_{ini}^2, \mathbf{x}, \mathbf{H})$ , where  $\delta_{ini}$  denotes an estimated initial sphere radius,  $\[]$  an empty control vector,  $J^2 = 0$  the initial cost and  $n = 1$  the first tree level.

---

#### Algorithm 1 Sphere Decoding algorithm

---

```

1: function SPHDEC( $\tilde{\mathbf{u}}, J^2, n, \delta^2, \mathbf{x}, \mathbf{H}$ )
2:   for each  $u \in \mathcal{U}$  do
3:      $\tilde{u}_n \leftarrow u$ 
4:      $J'^2 \leftarrow \|\mathbf{H}_{(n,1:n)}\tilde{\mathbf{u}}_{1:n} - x_n\|^2 + J^2$ 
5:     if  $\delta^2 \geq J'^2$  then
6:       if  $n < d$  then
7:         SPHDEC( $\tilde{\mathbf{u}}, J'^2, n + 1, \delta^2, \mathbf{x}, \mathbf{H}$ )
8:       else
9:          $\hat{\mathbf{u}} \leftarrow \tilde{\mathbf{u}}$ 
10:         $\delta^2 \leftarrow J'^2$ 
11:      end if
12:    end if
13:  end for
14:  return  $\hat{\mathbf{u}}$ 
15: end function

```

---

Traversal from one tree level to the next necessitates the consideration of all branching options or elements of the control set  $\mathcal{U}$  (line:2). The  $n^{th}$  component of the vector  $\tilde{\mathbf{u}}$  is updated in (line:3). Computation of the cost, i.e. squared distance between the target and the transformed vector  $\mathbf{H}\tilde{\mathbf{u}}$  in the  $n^{th}$ -dimension follows (line:4). Also, the accumulated cost of previous traversals  $J$ , are added. If the total cost  $J'^2 \leq \delta^2$  (line:5) and the algorithm is not yet at tree level  $n = d$  (line:6), the algorithm is called again to investigate branching to the next level. The search continues until  $J'^2 > \delta^2$ , in which case the algorithm prunes that specific branch and continues its search by considering the other branching options  $u \in \mathcal{U}$ . If all branchings are too expensive the algorithm steps back to the previous tree level and continues the search. If a path to a leaf node is discovered ( $n = d$ ), the leaf node is stored as a candidate solution (line:9) and the sphere radius is updated with the

reduced cost of traversal. The algorithm continues to search for less expensive paths until the tree is pruned to an extent where only one path remains. The corresponding leaf node is returned as the optimal solution  $\hat{\mathbf{u}}$ .

### 3.5.1.3 Computational complexity

The complexity of the SDA is quantified in terms of the number of operations required per node investigation multiplied by the number of nodes considered throughout the algorithm search [71]. Computations per node is a function of the dimensionality of the node under investigation. For the SDA listed in Algorithm 1, the computations per investigated node (line:4) total

$$n \otimes + n \oplus + 1 \ominus$$

element-wise operations. Note that  $\otimes$ ,  $\oplus$  and  $\ominus$  respectively denote an element-wise multiplication, addition and subtraction operation. Also, for *every iteration* of the algorithm, all elements of the set  $\mathcal{U}$  are considered. Hence, the inclusion of the control set's cardinality  $\#\mathcal{U}$  to give the total floating-point operations (flops)<sup>8</sup> per iteration as

$$\#\mathcal{U}(2n + 1) \text{ flops.} \quad (3.44)$$

This is a relatively low flop count, even if a full dimensional iteration  $n = d$  is considered. Complexity of the SDA arises from the size of the search tree, and therefore the number of nodes that are visited by the algorithm in different dimensions. This number is directly influenced by the sphere radius, and can be defined as the sum of the number of lattice points in spheres of radius  $\delta$  and dimensions  $n = 1, \dots, d$  [78]. An *optimal* sphere selection which contains only the optimal solution will effect a single path through the search tree, with a minimum of  $d$ -node visits (algorithm iterations). Such an idealistic selection sets the lower computational bound of Algorithm 1 for a three-level inverter at

$$\frac{3d^2 + 15d}{2} \text{ flops.} \quad (3.45)$$

Striving to keep the computational burden close to this lower bound necessitates the selection of an initial sphere radius with minimal length. The selection or estimation of

---

<sup>8</sup>A "floating-point operation" (flop) refers to any one floating-point operation (an add, subtract, multiply or divide). The *flop count* of a numerical algorithm has been used traditionally as an indicator of efficiency for solving a particular problem [96]. However, it is *not an accurate* predictor of *computation time* on modern computers, but is useful as a rough estimate of complexity [97].



such a radius is influenced by the *lattice structure*, i.e. the basis of matrix  $\mathbf{H}$ , and also the *location* of the target  $\mathbf{x}$  relative to the lattice. If the SDA is considered without any preconditioning and/or approximations, the upper computational bound can be extremely loose and, in a worst-case scenario, be exponential. Moreover, a couple of approaches exist which support the selection of a tight initial sphere radius. Particular to power electronic system, the sphere radius selection strategies of [39] and [48] dominate. Respectively, they employ an *educated guess* and a combination of the *educated guess*  $\tilde{\mathbf{u}}_{ed}$  and “so called” *Babai estimate*  $\tilde{\mathbf{u}}_{bab}$ . The educated guess is based on the assumption that elements of the optimal solution at the previous time-step  $\hat{\mathbf{u}}(k-1)$  shifted forward by one time-step, and repetition of the last control sequence renders a feasible solution. In accordance with the receding horizon principle, the educated guess can be obtained by the following process:

$$\tilde{\mathbf{u}}_{ed}(k) = \begin{bmatrix} \mathbf{0}_{d_u} & \mathbf{I}_{d_u} & \mathbf{0}_{d_u} & \dots & \mathbf{0}_{d_u} \\ \mathbf{0}_{d_u} & \mathbf{0}_{d_u} & \mathbf{I}_{d_u} & \ddots & \vdots \\ \vdots & \dots & \ddots & \ddots & \mathbf{0}_{d_u} \\ \mathbf{0}_{d_u} & \dots & \dots & \mathbf{0}_{d_u} & \mathbf{I}_{d_u} \\ \mathbf{0}_{d_u} & \dots & \dots & \mathbf{0}_{d_u} & \mathbf{I}_{d_u} \end{bmatrix} \hat{\mathbf{u}}(k-1).$$

Note that  $\mathbf{0}_{d_u}$  and  $\mathbf{I}_{d_u}$  respectively denote zero and identity matrices of a dimension equal to that of the control vector  $\mathbf{u}_{abc} \in \mathbb{Z}^{d_u}$  (2.45). From  $\tilde{\mathbf{u}}_{ed}(k)$  the initial sphere radius in the transformed  $\mathbf{H}$ -space is then computed from

$$\delta_{ed} = \|\mathbf{H}\tilde{\mathbf{u}}_{ed} - \mathbf{x}\|. \quad (3.46)$$

The Babai estimation [98] allows for computing an initial sphere radius which will enclose *at least* one lattice point in the sphere envelope. Computation of the Babai estimate is achieved by *rounding* the unconstrained minimum  $\tilde{\mathbf{u}}_{unc}(k)$  to the nearest integer vector

$$\tilde{\mathbf{u}}_{bab} = \lfloor \tilde{\mathbf{u}}_{unc} \rfloor. \quad (3.47)$$

The initial sphere radius in the transformed space is subsequently given by

$$\delta_{bab} = \|\mathbf{H}\tilde{\mathbf{u}}_{bab} - \mathbf{x}\|. \quad (3.48)$$

Combining these two approaches have shown to be quite effective in steady-state conditions where the CVP target resides inside or close to the span of the lattice. In [99] and

[100], the inverter switching constraint is included in the initial sphere radius estimation. This eliminates infeasibility issues that may arise from using the Babai estimate. The approach sequentially quantise each element of the vector  $\mathbf{u}_{unc}$  to the nearest integer in the control set (3.42c) while satisfying the switching constraint (2.49).

To improve on the *structure* of a lattice, lattice theory reverts to a process known as *lattice-basis-reduction*<sup>9</sup>. The process aims to define a more orthogonal lattice basis which gives way to better sphere radius estimations. Lattice reduction techniques may lead to some improvements in the solution of the ILS problem over *infinite* lattices, but it is not substantially useful for a subset of a lattice, as the lattice transforming matrix *often* destroys the lattice structure [78]. Lattice-basis-reduction also requires the computation of a transformation matrix, a process which in itself is NP-hard. The research presented in this thesis is interested in *finite* lattices and the *exact* solution to the optimisation problem underlying FCS-MPC. Therefore, lattice-basis-reduction techniques will not be considered in this thesis.

### 3.5.2 Iterative slicing algorithm

The “so-called” “Slicer” belongs to the category of exponential-space algorithms. In finding the closest lattice point to a vector, the algorithm presented in [101] achieves the objective by iteratively calculating which Voronoi cell of the lattice contains the given vector.

#### 3.5.2.1 Concept

The SDA finds the closest lattice point  $\mathbf{s} \in \mathbf{H}\tilde{\mathbf{u}}$  to a target vector  $\mathbf{x}$  through calculation of the Euclidean distance  $\|\mathbf{s} - \mathbf{x}\|$ . In comparison, the ISA finds the closest lattice point by calculating an error vector

$$\mathbf{e}_r = \mathbf{s} - \mathbf{x} \tag{3.49}$$

and then testing whether if the vector is contained in the basic Voronoi cell  $\mathbf{e}_r \in \mathcal{V}(\mathcal{L}, \mathbf{0})$ . In accordance with the translative nature of the basic Voronoi cell of a lattice, the error vector  $\mathbf{e}_r$  will reside inside the *basic Voronoi cell* if the target  $\mathbf{x}$  is closest to a specific

---

<sup>9</sup>Lattice reduction methods attempt to find an invertible, square  $d \times d$  transformation matrix  $\mathbf{T}$ , to the extent that  $\mathbf{T}$  and  $\mathbf{T}^{-1}$  have integer entries, and that the matrix  $\mathbf{G} = \mathbf{HT}$  is as orthogonal as possible [78].

lattice point  $\mathbf{s}$ . The name ‘‘Slicer’’ originates from the way in which the ISA verifies containment of  $\mathbf{e}_r \in \mathcal{V}(\mathcal{L}, \mathbf{0})$ . This is done through a number of *slicing* operations which require the precalculated set of ( $m = 2^d - 1$ ) single-sided Voronoi-relevant vectors  $\mathbf{V}$ , defined in (3.38). The basic Voronoi cell formulated in terms of inner products (3.39), is a combination of  $m$ -slices through  $\mathbb{R}^d$  space, each defined by a *single-sided* Voronoi-relevant vector  $\mathbf{v} \in \mathbf{V}$ . A single slice

$$\left\{ \mathbf{x} \in \mathbb{R}^d : |\langle \mathbf{x} \cdot \mathbf{v} \rangle| \leq \frac{1}{2} \|\mathbf{v}\|^2, \mathbf{v} \in \mathbf{V}, \mathbf{v} \neq \mathbf{0} \right\}, \quad (3.50)$$

is bounded by a pair of affine hyperplanes located at a distance of  $\pm \frac{1}{2} \mathbf{v}$  from the origin. Figure 3.9 illustrates the concept for the Voronoi-relevant vector  $\mathbf{v}_3$  in the two-dimensional example. Also note the dotted lines which borders translations of the original slice *centered* around at  $\pm(1, 2, \dots) \mathbf{v}_3$ .

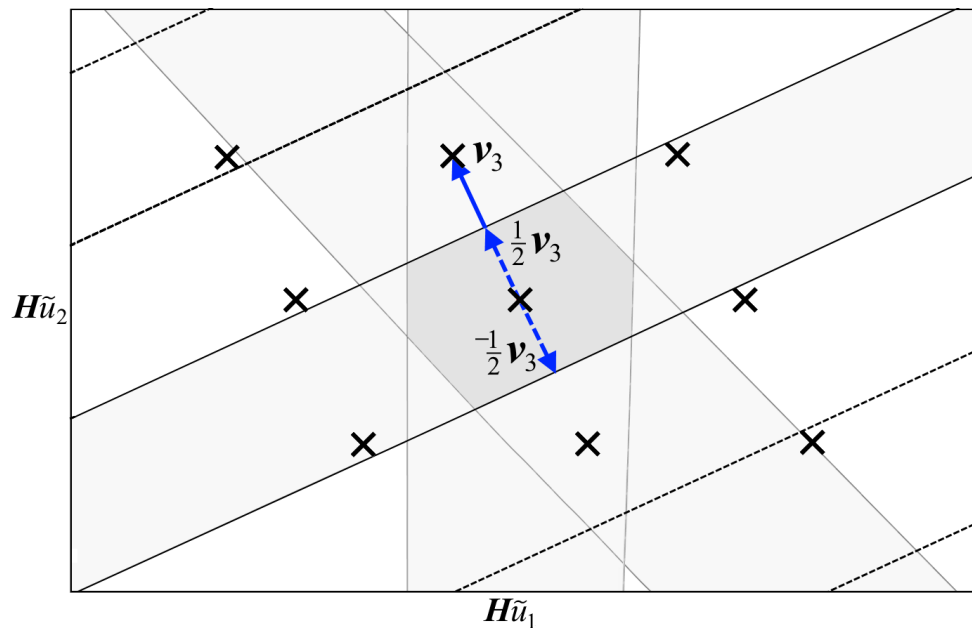


Figure 3.9: Slicing of the  $\mathbf{H}$ -coordinate space with Voronoi border pairs located at  $\pm \frac{1}{2} \mathbf{v}$ .

The idea behind the ISA is to start at some arbitrary lattice point  $\mathbf{s} = \mathbf{0}$ , and to shorten the error vector  $\mathbf{e}_r$  by subtracting Voronoi-relevant vectors in a sequential manner from  $\mathbf{s}$ . While  $\mathbf{e}_r$  remains outside the basic Voronoi cell, the algorithm progresses and updates  $\mathbf{s}$  until  $\mathbf{e}_r$  is found to reside within *all* the slices.

In the following Figures (Fig. 3.10 to Fig. 3.13) the slicing process is demonstrated by means of a two-dimensional CVP example. Fig. 3.10 shows the target  $\mathbf{x}$ , some arbitrary

starting lattice point  $\mathbf{s}$  and the error vector  $\mathbf{e}_r$  as a result of (3.52). The ISA considers the Voronoi-relevant vectors in some sequential manner, for instance, we choose  $\mathbf{v}_3$  as the first option. Projection of  $\mathbf{e}_r$  onto  $\mathbf{v}_3$  (Fig. 3.11) results in a projection coefficient (3.40b) with magnitude larger than 1.5. This identifies  $\mathbf{e}_r$  as an element of the slice defined by  $\pm\frac{1}{2}\mathbf{v}_3$  and translated with  $k\mathbf{v}_3$ ,  $k = -2$ .

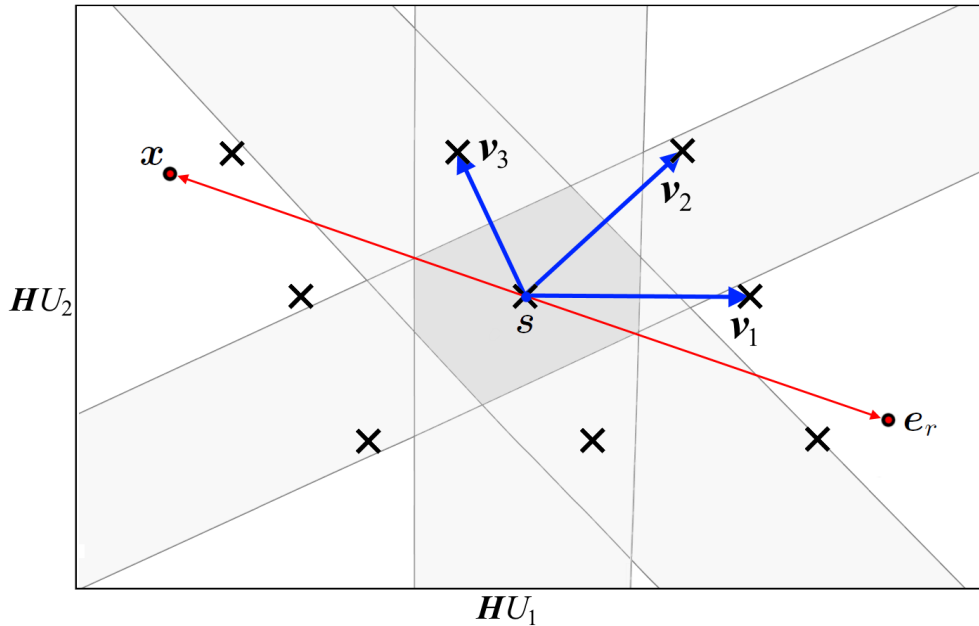


Figure 3.10: The error vector  $\mathbf{e}_r$  as a result of the target  $\mathbf{x}$  and a starting lattice point ( $\mathbf{s} = \mathbf{0}$ ).

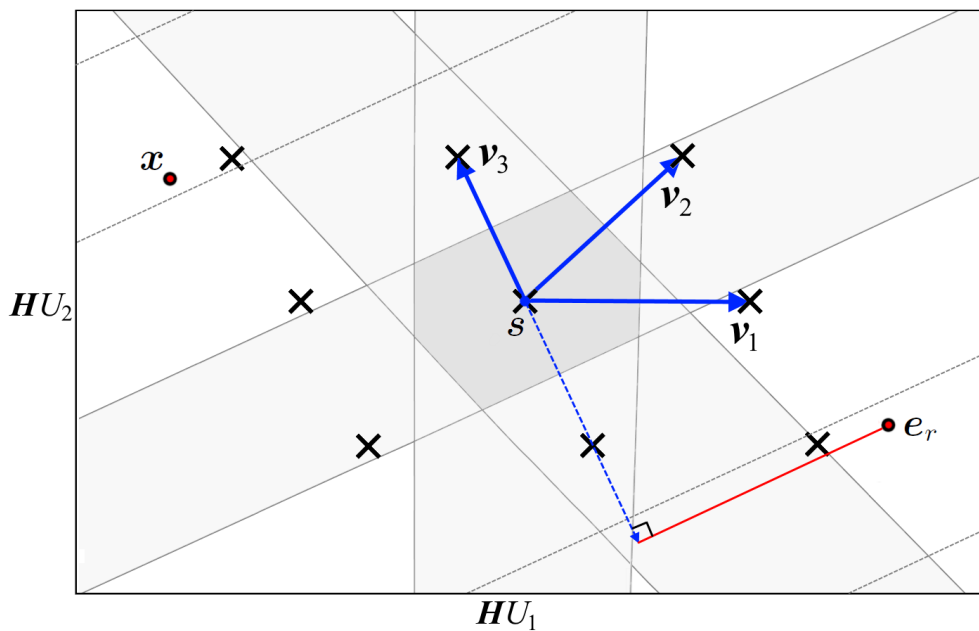


Figure 3.11: Projection of  $\mathbf{e}_r$  onto  $\mathbf{v}_3$  resulting in a projection coefficient with a magnitude larger than 1.5

With the  $k^{\text{th}}$  slice identified, the lattice point  $\mathbf{s}$  is updated with  $-k\mathbf{v}_3$ . Figure 3.12 shows the updated lattice point  $\mathbf{s} = 2\mathbf{v}_3$  and the corresponding error vector  $\mathbf{e}_r$ , which now resides within the slice defined by  $\pm\frac{1}{2}\mathbf{v}_3$ . The next Voronoi-relevant vector  $\mathbf{v}_2$  is considered by projecting  $\mathbf{e}_r$  onto it. A projection coefficient larger than 0.5, but less than 1.5, identifies  $\mathbf{e}_r$  as an element of the slice defined by  $\pm\frac{1}{2}\mathbf{v}_2$  and translated with  $k\mathbf{v}_2$ ,  $k = 1$ . Updating  $\mathbf{s}$  with  $-k\mathbf{v}_2$  results in the lattice point and corresponding error vector as indicated in Fig. 3.13.

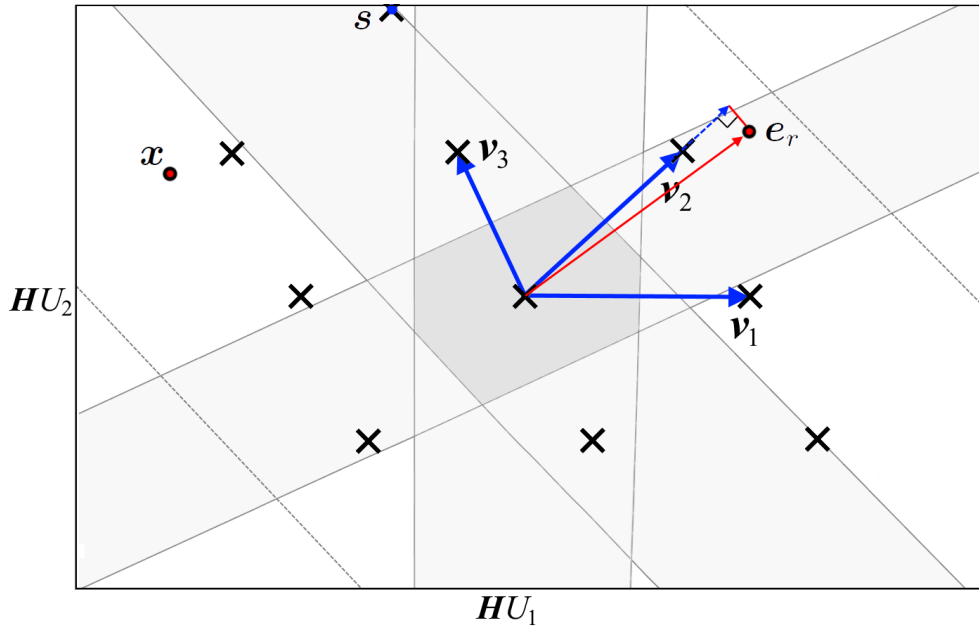


Figure 3.12: Updating the lattice point gives  $\mathbf{s} = \mathbf{0} + 2\mathbf{v}_3$ , with the new error vector residing in slice  $\pm\frac{1}{2}\mathbf{v}_3$ . Projection of  $\mathbf{e}_r$  onto  $\mathbf{v}_2$  results in a projection coefficient with a magnitude larger than 0.5, but less than 1.5.

Note that  $\mathbf{e}_r$  now resides within the slices defined by both  $\pm\frac{1}{2}\mathbf{v}_3$  and  $\pm\frac{1}{2}\mathbf{v}_2$ . Projection of  $\mathbf{e}_r$  onto the remaining Voronoi-relevant vector  $\mathbf{v}_1$  effects a projection coefficient of less than 0.5, which subsequently confirms containment in the slice defined by  $\pm\frac{1}{2}\mathbf{v}_1$ . As a result, the error vector is contained in the basic Voronoi cell that identifies  $\mathbf{s}$  as the closest lattice point to the target vector, and therefore the solution to the CVP

$$\hat{\mathbf{s}} = \mathbf{x} + \mathbf{e}_r. \quad (3.51)$$

### 3.5.2.2 Algorithm

The pseudo code for the Iterative slicing algorithm is listed as Algorithm 2. Upon initiation it requires the precalculated set of single-sided Voronoi vectors  $\mathbf{V}$  and generator

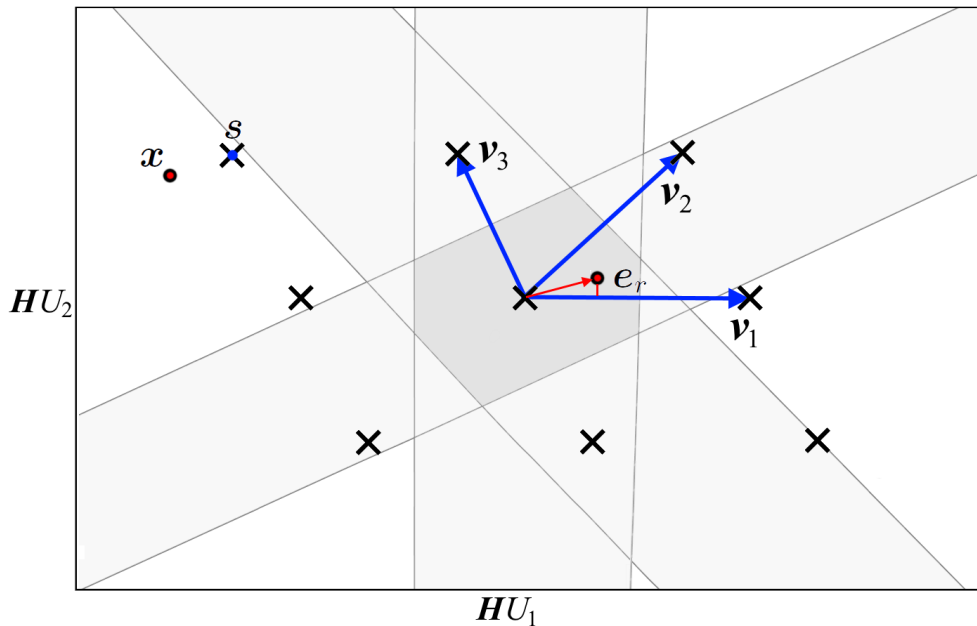


Figure 3.13: Updating the lattice point gives  $\mathbf{s} = \mathbf{0} + 2\mathbf{v}_3 - 1\mathbf{v}_2$  with the effected error vector residing in slices  $\pm\frac{1}{2}\mathbf{v}_3$  and  $\pm\frac{1}{2}\mathbf{v}_2$ . Projection of  $\mathbf{e}_r$  onto  $\mathbf{v}_1$  results in a projection coefficient with a magnitude less than 0.5, signifying containment by the slice  $\pm\frac{1}{2}\mathbf{v}_1$ .

matrix  $\mathbf{H}$ . It typically starts with the lattice point at the origin  $\mathbf{s} = \mathbf{0}$ , and as a first step computes the error vector (line:7). A *special condition* is tested and adjusted for in (lines: 6 to 8); this will be discussed at a later stage. In (line:10) the error vector is projected onto the first Voronoi vector  $\mathbf{v}_j$  and rounded with the function  $rnd$ . This function differs from the normal rounding operation, in that the midpoint between successive integers is rounded to the integer with smaller absolute value, i.e.  $rnd(\pm 0.5) = 0$ . This allows for the result  $k \in \mathbb{Z}$ , to define in which  $k$ -slice, of  $\mathbf{v}_j$  the error vector resides. If  $k = 0$  then  $\mathbf{e}_r$  resides in the original or 0-slice, and no adjustment is required. The current lattice point  $\mathbf{s}$  is updated with  $-\mathbf{k}\mathbf{v}_j$  (line:14). This effects the updated error vector  $\mathbf{e}_r$  to reside in the 0-slice of the current  $\mathbf{v}_j$ . This process is repeated for each  $\mathbf{v}_j$  until the counter equals  $m$ , i.e.  $\mathbf{e}_r$  reside within *all*  $m$ -slices  $\mathbf{e}_r \in \mathcal{V}(\mathcal{L})$ . Upon achieving this objective, the most recent lattice point  $\mathbf{s}$  is identified as the closest to  $\mathbf{x}$ .

At a glance, the search process seems to be exhaustive in nature as *all* the Voronoi-relevant vectors are investigated and tested in a sequential manner. However, the algorithm implements two principles for speeding up the decoding process. Firstly, in the precalculation stage, the Voronoi-relevant vectors are sorted in an *ascending order* in

**Algorithm 2** Iterative Slicing algorithm

---

```

1: function ISA( $\mathbf{x}, \mathbf{s}, \mathbf{V}, \mathbf{H}$ )
2:    $cnt \leftarrow 0$ 
3:    $j \leftarrow 0$ 
4:   while  $cnt < m$  do
5:      $\mathbf{e}_r = \mathbf{s} - \mathbf{x}$ 
6:     if  $\|\mathbf{e}_r\|^2 < 0.25 \|\mathbf{v}_j\|^2$  then
7:        $cnt = cnt + m - j$ 
8:        $j = m - 1$ 
9:     else
10:       $k = \text{rnd}(\langle \mathbf{v}_j \cdot \mathbf{e}_r \rangle / \|\mathbf{v}_j\|^2)$ 
11:      if  $k = 0$  then
12:         $cnt = cnt + 1$ 
13:      else
14:         $\mathbf{s} = \mathbf{s} - k\mathbf{v}_j$ 
15:         $cnt = 0$ 
16:      end if
17:       $j = j + 1, (\text{mod } m)$ 
18:    end if
19:  end while
20:  return  $\hat{\mathbf{u}} = \mathbf{H}^{-1}\mathbf{s}$ 
21: end function

```

---

terms of their squared Euclidean norms

$$\|\mathbf{v}_1\|^2 \leq \|\mathbf{v}_2\|^2 \leq \dots \leq \|\mathbf{v}_m\|^2. \quad (3.52)$$

This arrangement bears results during *online* operation when the error vector is evaluated for the *special condition* (line: 6 to 8) we overlooked before. The statement verifies that the norm of the current error vector is less than half the norm of the current Voronoi-relevant vector, i.e.

$$\|\mathbf{e}_r\|^2 < 0.25 \|\mathbf{v}_j\|^2 \equiv \|\mathbf{e}_r\| < 0.5 \|\mathbf{v}_j\|. \quad (3.53)$$

If the statement holds, there is no need to evaluate containment of  $\mathbf{e}_r$  in the slice of  $\mathbf{v}_j$  and all the other slices of  $\mathbf{v}_{(j+1), \dots, m}$ , as  $\mathbf{v}_j \in \mathbf{V}$  is ranked in an ascending order (3.52). Secondly, further improvement is obtained by initialising the algorithm from a lattice point closer to the given target vector, as opposed to the typical lattice point  $\mathbf{s} = \mathbf{0}$ . The improved starting point will ensure a smaller error vector, resulting in less iterations and faster termination of the algorithm. Sommer(2009) [101] suggested that such an improved starting point can be found from obtaining the rounded least-squares solution

$$\mathbf{s} = \mathbf{H} \lfloor \mathbf{H}^{-1}\mathbf{x} \rfloor. \quad (3.54)$$

### 3.5.2.3 Computational complexity

The most computationally demanding part of the algorithm is the calculation of  $k$  in (line:10), which involves vector operations and is performed at every step. As part of preprocessing the squared norms  $\|\mathbf{v}_j\|^2$  of the Voronoi-relevant vectors and their reciprocals  $\|\mathbf{v}_j\|^{-2}$  are calculated offline. This leaves the dot products and multiplications (lines: 6 and 10) for online calculation. Together with the computation of  $\mathbf{e}_r$  (line:5) and updating of  $\mathbf{s}$  in (line:14), the computational cost per investigated Voronoi-relevant vector amounts to

$$2(d \otimes + (d - 1) \oplus + \otimes) + 2d \ominus + d \otimes$$

element-wise operations. Since a maximum of  $(2^d - 1)$  Voronoi-relevant vectors can exist, the complexity of one *full iteration* is given by

$$7d(2^d - 1) \text{ flops.} \quad (3.55)$$

The added  $(2^d - 1)$  complexity unfortunately results in the ISA being inferior to the SDA's performance (3.44) in large dimensions [101]. Also with the ISA being an iterative process, the number of lattice points to investigate (algorithm iterations) has the potential to be extremely high. Therefore, the ISA unfortunately do not perform any better than previous CVP algorithms, except that it terminates after a finite number of iterations [46].

## 3.5.3 Micciancio Voulgaris algorithm

The algorithm presented by Micciancio and Voulgaris, commonly referred to as the MV algorithm (MVA) [46], solves the closest vector problem with preprocessing (CVPP) by employing the Voronoi cell of a lattice as a result of the preprocessing function. It is based on the Iterative Slicing algorithm, and addresses the main shortfall of the ISA in terms of the manner in which  $\mathbf{s}$  is updated with  $\mathbf{v}$  until a solution is obtained.

### 3.5.3.1 Concept

Similar to the ISA is an error vector  $\mathbf{e}_r$  (3.49) calculated and subjected to a containment test in terms of the basic Voronoi cell  $\mathcal{V}$ . If  $\mathbf{e}_r \notin \mathcal{V}$ , the lattice point  $\mathbf{s}$  is updated with a Voronoi-relevant vector  $\mathbf{v}$ , effectively shortening  $\mathbf{e}_r$  to conclusively effect residency in



$\mathcal{V}$ . What sets the MVA apart from the ISA is a novel selection strategy which identifies an *optimum Voronoi-relevant vector*  $\mathbf{v}^*$  to update  $\mathbf{s}$  with. The optimum Voronoi-relevant vector is selected from the set of single-sided Voronoi-relevant vectors  $\mathbf{V}$ , as the vector with the *largest absolute* projection coefficient, i.e.

$$\mathbf{v}^* = \arg \max_v |k_v| \quad \text{where} \quad k_v = \frac{\langle \mathbf{e}_r, \mathbf{v} \rangle}{\|\mathbf{v}\|^2} \quad \text{for all } \mathbf{v} \in \mathbf{V}, \quad \mathbf{v} \neq \mathbf{0}. \quad (3.56)$$

This strategy relates to the concept known as the *dual interpretation of projection*<sup>10</sup> by substituting the measure of maximum Euclidean distance with the largest projection coefficient. Geometrically, the projection coefficients  $k_v$ , can be interpreted as indicators to where the error vector resides in terms of the individual Voronoi-relevant vectors. The optimum vector  $\mathbf{v}^*$  with maximum  $k$ -value suggests the strongest directional component, which will effect an optimal reduction of  $\mathbf{e}_r$  when updating  $\mathbf{s}$  with  $\mathbf{v}^*$ . Figure 3.14 illustrates the concept with some  $\mathbf{e}_r = \mathbf{s} - \mathbf{x}$  projected orthogonally onto the Voronoi-relevant vectors. From the orthogonal projections it can be observed that for  $\mathbf{v}_1$  and  $\mathbf{v}_2$ ,  $k$ -values

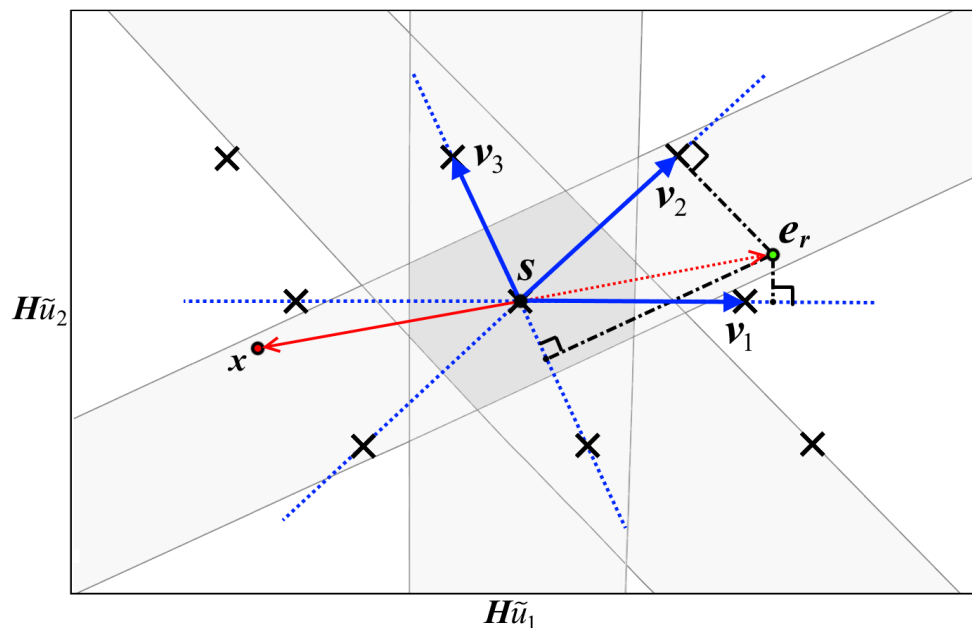


Figure 3.14: Orthogonal projection of  $\mathbf{e}_r$  onto the set of Voronoi-relevant vectors to identify the optimum vector  $\mathbf{v}^* = \mathbf{v}_2$ .

will be larger than a 0.5 result whereas for  $\mathbf{v}_3$  will  $k_{\mathbf{v}_3} < 0.5$ . This stipulates that  $\mathbf{e}_r$  is outside the slices of  $\mathbf{v}_1$  and  $\mathbf{v}_2$  but inside the slice of  $\mathbf{v}_3$ . The largest projection coefficient

<sup>10</sup>Minimum distance projection of a point  $\mathbf{x}$  onto a convex hull will result in a projection vector with maximum Euclidean distance from  $\mathbf{x}$  to all the hyperplanes supporting the convex hull and separating it from  $\mathbf{x}$ . This concept is referred to as the dual interpretation of projection [102, p.134], [97, p.602]. A detailed description of the interpretation is given in Appendix A.6.

results from the projection onto  $\mathbf{v}_1$ , as it obviously poses the strongest directional component relative to  $\mathbf{e}_r$ . Updating  $\mathbf{s} = \mathbf{s} - \mathbf{v}_1$  results in a new error vector  $\mathbf{e}_r = \mathbf{s} - \mathbf{x}$  that resides within all the slices (see Fig. 3.15). With  $\mathbf{e}_r \in \mathcal{V}$ , the closest lattice point to  $\mathbf{x}$  is

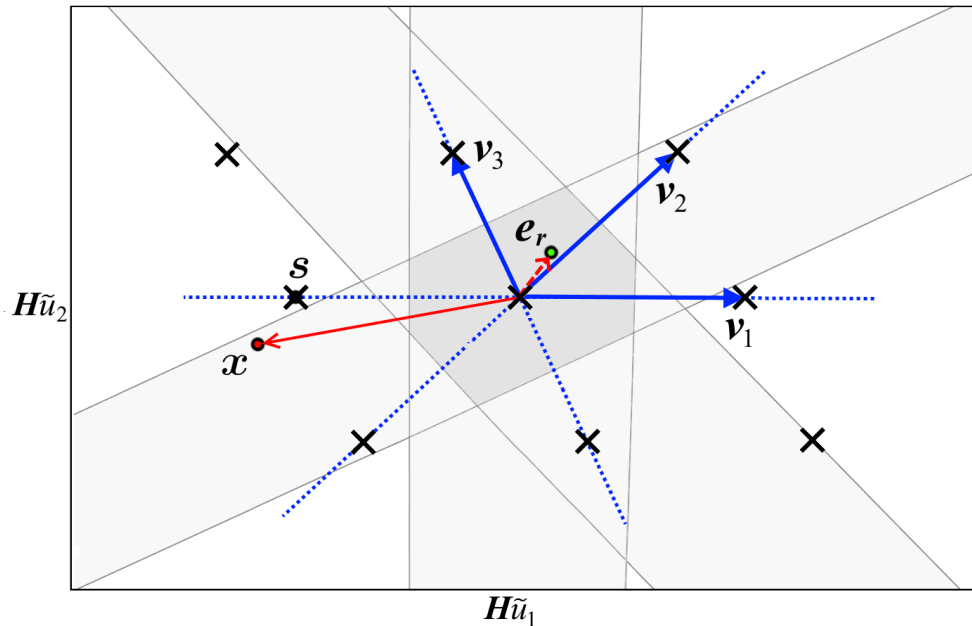


Figure 3.15: Updating the lattice point  $\mathbf{s}$  with the optimum Voronoi-relevant vector  $-\mathbf{v}_1$ , resulting in the error vector  $\mathbf{e}_r \in \mathcal{V}$ .

the most recent lattice point  $\hat{\mathbf{s}}$ , which satisfies the equality

$$\hat{\mathbf{s}} = \mathbf{x} + \mathbf{e}_r. \quad (3.57)$$

### 3.5.3.2 Algorithm

Algorithm 3 lists the *Micciancio Voulgaris* CVPP algorithm from [46], presented in terms of the ISA variables for the purpose of demonstrating the improved selection strategy. As

---

#### Algorithm 3 *Micciancio Voulgaris* CVP algorithm

---

```

1: function MVA( $\mathbf{x}, \mathbf{s}, \mathbf{V}, \mathbf{d}, \mathbf{H}$ )
2:    $k^* \leftarrow 1$ 
3:   while  $k^* \neq 0$  do
4:      $\mathbf{e}_r = \mathbf{s} - \mathbf{x}$ 
5:      $\mathbf{k} = \mathbf{V}^T \mathbf{e}_r \otimes \mathbf{d}$ 
6:      $j = \arg \max |\mathbf{k}|$ 
7:      $k^* = \text{rnd}(k_j)$ 
8:      $\mathbf{s} = \mathbf{s} - k^* \mathbf{v}_j$ 
9:   end while
10:  return  $\hat{\mathbf{u}} = \mathbf{H}^{-1} \mathbf{s}$ 
11: end function

```

---

inputs the MVA requires the target vector  $\mathbf{x}$ , an initial lattice point  $\mathbf{s}$  and the set of pre-calculated, *single-sided* Voronoi-relevant vectors  $\mathbf{V}$  (3.38). The reciprocals of the squared norms of the respective Voronoi-relevant vectors are also precalculated and presented as elements of the vector

$$\mathbf{d} = [\|\mathbf{v}_1\|^{-2} \ \|\mathbf{v}_2\|^{-2} \ \dots \ \|\mathbf{v}_m\|^{-2}]^T. \quad (3.58)$$

Similar to the ISA, the MVA can be initiated with  $\mathbf{s} = \mathbf{0}$ , or the rounded least-squares solution (3.54). The algorithm starts by setting  $k^* \neq 0$ , followed by the calculation of the  $\mathbf{e}_r$ . Orthogonal projection of  $\mathbf{e}_r$  onto the set of Voronoi-relevant vectors  $\mathbf{V}$  follows to produce the respective projection coefficients as elements of

$$\mathbf{k} = [k_1 \ k_2 \ \dots, k_m]^T. \quad (3.59)$$

The  $j^{\text{th}}$  element of  $\mathbf{k}$  with maximum absolute value is determined (line:6). As per the definition of the Voronoi cell (3.39) and the subsequent containment test (3.41), the *rnd* function (defined in the ISA) will result in  $k^* = 0$  if  $\mathbf{e}_r \in \mathcal{V}$ . If this is not the case, then  $\mathbf{s}$  is updated by subtracting  $k^*$ -times the *optimal* Voronoi-relevant vector  $\mathbf{v}_j$ . The algorithm repeats until  $k^* = 0$ , in which case the containment test (3.41) is satisfied with the most recent lattice point  $\mathbf{s}$ , identifying the closest lattice point to  $\mathbf{x}$ . The solution to the optimisation problem is lastly returned as  $\hat{\mathbf{u}} = \mathbf{H}^{-1}\mathbf{s}$ .

### 3.5.3.3 Computational complexity

In solving the CVP, every iteration of Algorithm 3 (lines 4 to 8) requires

$$(2^d - 1) ((d + 1) \otimes + (d - 1) \oplus) + 2d \ominus + d \otimes \quad (3.60)$$

element wise computations, which totals to

$$(2^{d+1} + 1)d \text{ flops.} \quad (3.61)$$

The special case where the target vector is guaranteed to belong to twice the Voronoi cell, bounds the computational sequence to  $2^d$  iterations [46]. This constitutes a time complexity of  $\mathcal{O}2^{2d+1}$ , i.e. single exponential time.

### 3.5.3.4 Challenge of lattice truncation

The MVA solves the CVP by determining which Voronoi cell of the lattice contains the target vector. In the FCS-MPC of a converter, the finite control set effects a truncated

lattice. For the MVA it is essential that the target resides in a translation of the basic Voronoi cell. In Section 3.2 it was highlighted that the Voronoi regions of the lattice points spanning the outer border of the *truncated* lattice are unbounded and thus *not closed Voronoi cells*. As a consequence, whenever the target resides far away from the lattice, it will not be contained in a tiling of the basic Voronoi cell. In such a case it is suggested by Micciancio and Voulgaris(2013) [46] to simply project the target  $\mathbf{x}$  onto the lattice and utilise the projected point as an updated target  $\tilde{\mathbf{x}}$ . This will ensure containment of  $\tilde{\mathbf{x}}$  by a tiling of the basic Voronoi cell. Figure 3.16 demonstrates the issue of a distant target and the proposed solution thereto. This concept, particular to the application of the MVA as solver in an FCS-MPC power electronic system, was reported on in [103] and will be revisited in the next section.

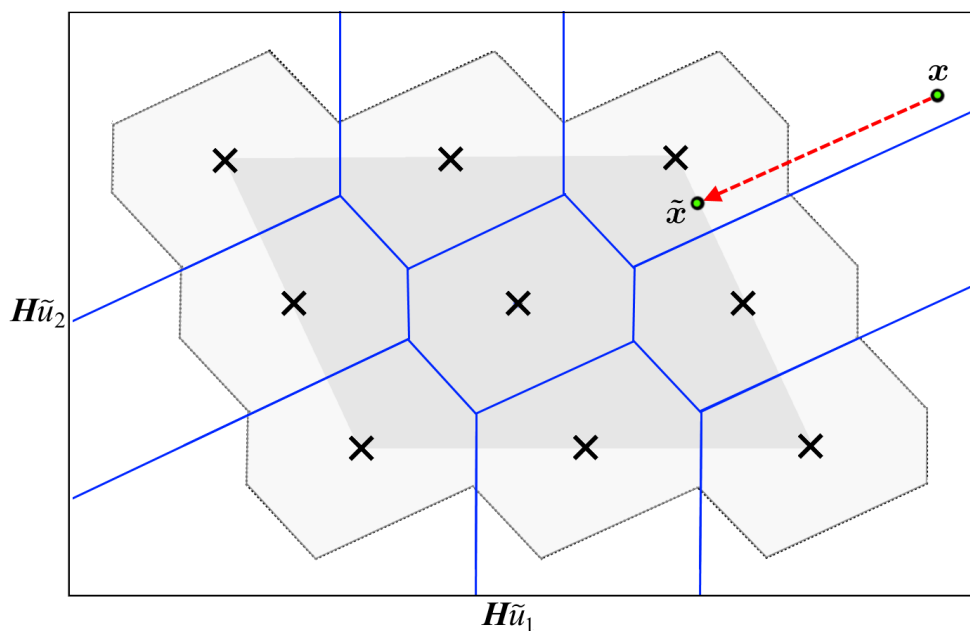


Figure 3.16: Truncated lattice with target residing outside translations of the basic Voronoi cell and proposed orthogonal projection.

### 3.6 Conditioning of the CVP target

The process of finding a point  $\tilde{\mathbf{x}}$  in a convex set  $\mathcal{P}$  that is closest to an arbitrary point  $\mathbf{x} \in \mathbb{R}^d$  in an Euclidean sense, is a convex optimisation problem which is commonly referred to as *minimum distance projection* [97, p.88]. In [104] the concept was employed to find a constrained solution from an unconstrained solution for applications where a finite number of control actions are available. In such cases where a finite lattice is of

concern, the integrity of the projection process stems from the fact that the Voronoi border between two adjacent Voronoi sites are orthogonal to the Voronoi-relevant vector that connects them. This supports the concept that the Voronoi border separating two lattice points in the *outer bound* of the lattice, is perpendicular to the lattice convex hull. Orthogonal projection onto the lattice convex hull will as a result maintain integrity, as no Voronoi borders will be crossed in the projection process. The projection ending (on the convex hull) will thus remain in the same Voronoi cell as where it originated from (see Fig. 3.16). However, in some instances where the lattice basis poses a high orthogonality defect, the optimality of this projection approach can be affected. To simplify explanation of the projection process, the optimality concern will initially be disregarded and later be reintroduced in Subsection 3.6.4.

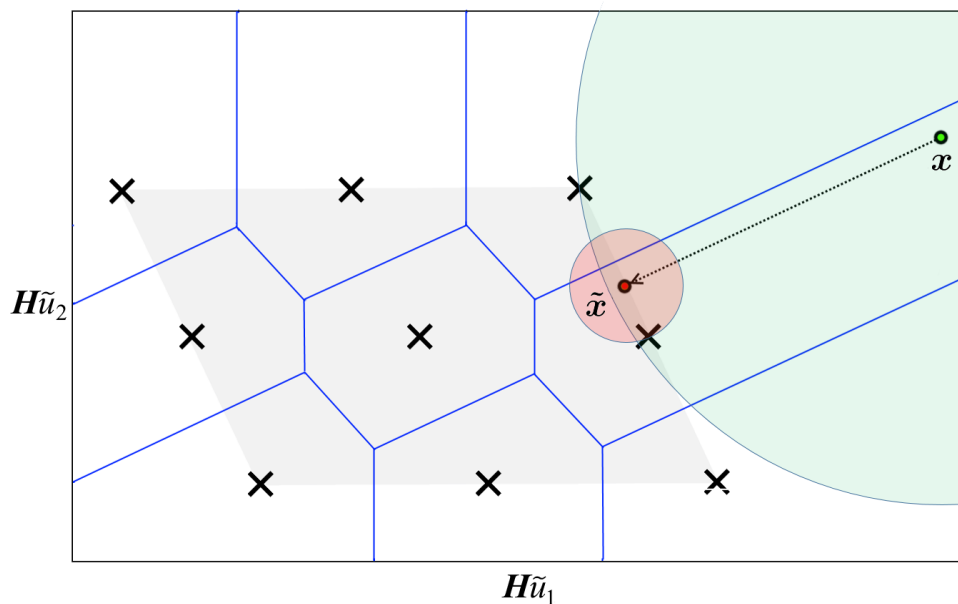


Figure 3.17: Effect of orthogonal projection onto the convex hull of the lattice.

Specific to FCS-MPC of power electronic systems, the issue of the CVP target being distant to the lattice search space was reported in [105] to occur during transient operating conditions of the system under control. In [105], the SDA was used as solver, with the effect on the initial sphere-radius demonstrated. A solution to the issue was proposed in the form of orthogonal projection onto the lattice convex hull. Figure 3.17 illustrates the effect on initial sphere radius before and after the projection process. In both instances, the Babai estimation (3.48) is considered. In [106] the computational issue as a result of

the inflated sphere radius during transient events was also investigated. Again, orthogonal projection was proposed to alleviate the computational burden.

The projection of a point onto a  $d$ -dimensional parallelotope is not a trivial matter and bears a significant computational requirement, which adds to the total burden of the decoding process. To economise the computational cost of the projection problem, both [105] and [106] considered it as a quadratic problem which is box-constrained in the original or input space. Optimisation of the QP was achieved with off-the-rack solvers such as the interior point or gradient projection methods. In both these papers extensive improvement in the SDA's performance was demonstrated. However, a minimal loss in optimality due to projection under certain conditions was reported.

### 3.6.1 Proposed projection algorithm

In the research presented here an alternative projection algorithm is developed which is tailored specific for the search spaces originating from FCS-MPC problems for *three-phase* multilevel inverters. By focusing on the optimisation problem unique to these systems, a computationally efficient algorithm is proposed. The algorithm presented here solves the projection problem in the transformed space by means of geometrical optimisation.

MPC of a three-level inverter with control set  $\mathcal{U}$  per inverter-arm (2.42) effects an input space which resembles a truncated hyper-cubic lattice. The convex hull of this lattice can be interpreted a  $d$ -dimensional box or *hypercube* (3.9). Construction of a hypercube follows a simple path; it starts as a single point (0-cube) and evolves into higher dimensions through the process of *sweeping out volumes* (see Appendix A.5). A three-dimensional cube (3-cube) might be considered a simple object, however in higher dimensions, the number of faces describing a full-dimensional hypercube is quite daunting (see the table in Appendix A.5). The *linear mapping*<sup>11</sup> of a hypercube with the square matrix  $\mathbf{H}$  will effect transformation (rotation, scaling and/or reflection) to result in a *parallelotope* in a new coordinate space of equal dimension. Congruently, the construction of the parallelotope can be considered in terms of the sweeping out of volumes. Starting

<sup>11</sup>A linear transformation  $\mathbf{T} : \mathbb{R}^d \rightarrow \mathbb{R}^d$  that maps within the same  $d$ -dimensional space associates  $\mathbf{T}$  with a  $(d \times d)$ -matrix  $\mathbf{T}$ , to such an extent that  $\mathbf{T}(\mathbf{x}) = \mathbf{H}\mathbf{x}$  [107]. The linear transformation of a  $d$ -dimensional parallelotope in a  $d$ -dimensional space will always result in some type of  $d$ -dimensional parallelotope [108].

from a 0-parallelotope (point), sweeping out with the first basis vector  $\pm \mathbf{h}_1$  gives a 1-parallelotope (line-segment), sweeping out the line-segment with  $\pm \mathbf{h}_2$  constitutes a 2-parallelotope (parallelogram), sweeping out the parallelogram with  $\pm \mathbf{h}_3$  constitutes a 3-parallelotope (parallelepiped), and so the process continues into higher dimensions.

### 3.6.2 Concept

The minimum distance projection from a point  $\mathbf{x} \in \mathbb{R}^d$  to a convex set  $\mathcal{P} \in \mathbb{R}^d$  can be found by *maximising the distance* from  $\mathbf{x}$  to a hyperplane  $\mathcal{H}$  over the set of *all* hyperplanes *separating*<sup>12</sup>  $\mathbf{x}$  from  $\mathcal{P}$  [102]. This concept is also referred to as the *dual interpretation of projection* [109] (see Appendix A.6), and forms the basis of the proposed projection algorithm. The objective is to find the *optimal separating hyperplane* which is characterised by the fact that it also supports the convex set  $\mathcal{P}$ .

Consider the two dimensional ( $d = 2$ ), three-level ( $l_v = 3$ ) lattice  $\mathcal{L}(\mathbf{H})$  (3.12) with a convex hull  $\mathcal{P}(\mathcal{L}(\mathbf{H}))$  (3.27) depicted in Figure 3.18. Note that the convex hull  $\mathcal{P} \in \mathbb{R}^d$

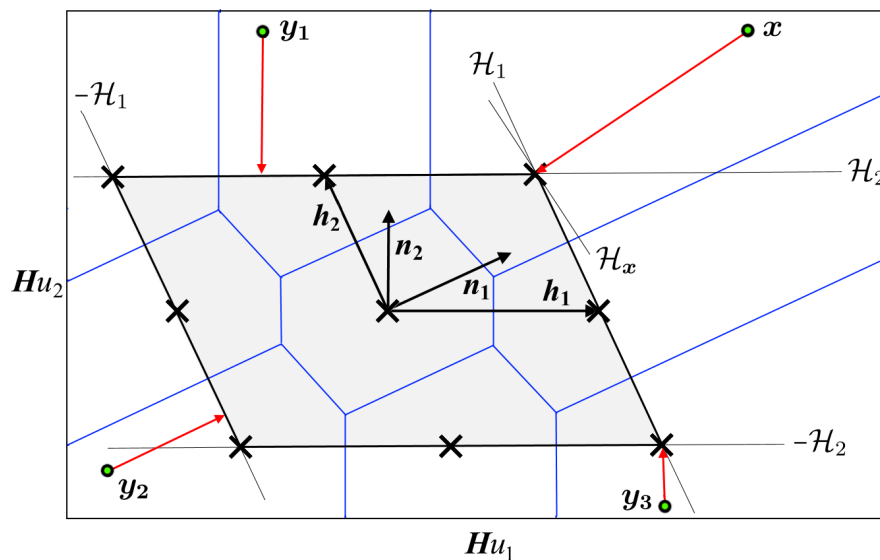


Figure 3.18: Minimum distance projection vector onto a convex hull in  $d$ -dimensional space.

is supported by the set of  $(d - 1)$ -dimensional hyperplanes  $\pm \mathcal{H}$  (3.14) which contain the  $2d, (d - 1)$ -dimensional faces (lines) of the parallelotope. Where the  $(d - 1)$ -dimensional

<sup>12</sup>The hyperplane separation theorem is obeyed when two nonempty convex sets in a  $d$ -dimensional Euclidean space are separated by a hyperplane. *Separation* implies that each set belongs to a half-space on opposing sides of the hyperplane [97].

hyperplanes intersect, the  $2^d$ ,  $(d - 2)$ -dimensional faces (points or vertices) of the convex hull are formed. The minimum distance projections originating from  $\mathbf{y}_{1,\dots,3}$  onto the convex hull are relatively simple and easy to compute as they are perpendicular to one of the  $(d - 1)$ -dimensional hull hyperplanes which is partial to the convex hull. In comparison, the projection originating from  $\mathbf{x}$  shows the  $(d - 2)$ -dimensional vertex as the closest point on the convex hull as an element of some *optimal* separating hyperplane  $\mathcal{H}_x$ . Finding the  $(d - 1)$ -dimensional hyperplane  $\mathcal{H}_x$  in  $d$ -dimensional space is a challenging task, especially in higher dimensions. To simplify the projection process, the proposed algorithm employs the concept of *orthogonal decomposition*<sup>13</sup> of a vector. Figure 3.19 shows the decomposition of the projection vector  $\hat{\mathbf{p}}$  into a duo of orthogonal vectors. The

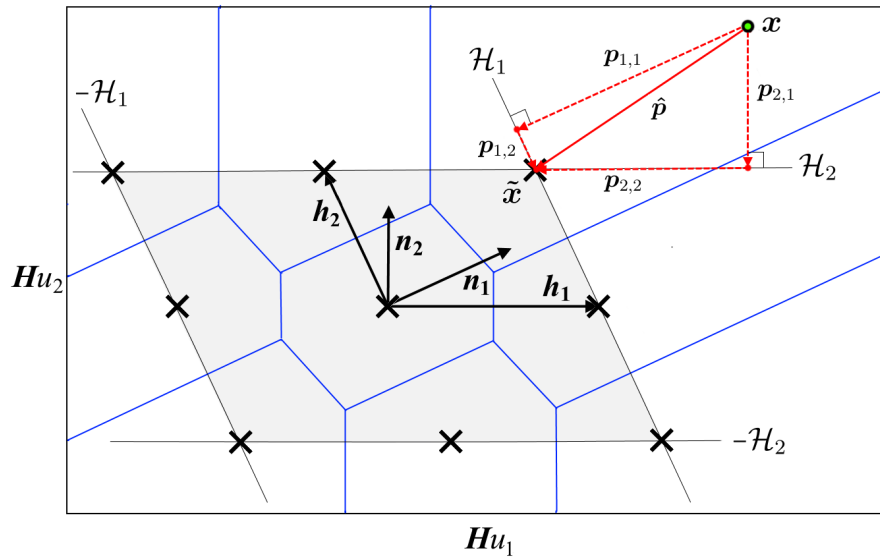


Figure 3.19: Orthogonal decomposition of the projection vector in a  $d$ -dimensional space.

one vector  $\mathbf{p}_{1,1}$  is the orthogonal projection of  $\mathbf{x}$  onto the affine hyperplane  $\mathcal{H}_1$ , and the other a vector  $\mathbf{p}_{1,2}$  which is *in* the affine hyperplane  $\mathcal{H}_1$ . Combined, the duo of vectors result in  $\hat{\mathbf{p}} = \mathbf{p}_{1,1} + \mathbf{p}_{1,2}$ . Similarly, it can be observed that, if the affine hyperplane  $\mathcal{H}_2$  is considered, the same vector can alternatively be decomposed as  $\hat{\mathbf{p}} = \mathbf{p}_{2,1} + \mathbf{p}_{2,2}$ .

To *ensure optimality*, the algorithm adheres to the *dual interpretation of projection* by identifying a hull hyperplane which is closest to being the *optimal separating hyperplane*, i.e. at a *maximum* orthogonal distance from  $\mathbf{x}$ . For the example, hyperplane  $\mathcal{H}_1$  is identified as the stronger separating hyperplane due to the magnitude of projection vector

<sup>13</sup>see Appendix A.6 for a detailed explanation.



$\mathbf{p}_{1,1}$ . It is observed, that the  $(d-1)$ -dimensional hull-face (line-segment) and two  $(d-2)$ -dimensional faces (vertices) of the  $d$ -parallelotope are contained in the affine space spanned by  $\mathcal{H}_1$ . Computation of vector  $\mathbf{p}_{1,2}$  requires the *minimum distance projection* from the point  $\mathbf{x} + \mathbf{p}_{1,1}$  onto the relevant  $(d-1)$ -dimensional hull-face. With both the projection vector  $\mathbf{p}_{1,2}$  and the hull-face residing in the  $(d-1)$ -dimensional affine space of  $\mathcal{H}_1$ , the projection process is simplified by translating the affine space to a subspace  $\mathcal{H}_{1,0}$ . This effectively reduces or sweeps in the  $d$ -parallelotope to effect a  $(d-1)$ -parallelotope (line-segment), which is supported by two  $(d-2)$ -dimensional affine hull-hyperplanes (vertices) located at  $\pm \mathbf{h}_2$ . Figure 3.20 illustrates the  $(d-1)$ -dimensional subspace  $\mathcal{H}_{1,0}$  and the resident parallelotope spanned by  $\pm \mathbf{h}_2$ . Note that the hyperplanes supporting the new parallelotope hull are defined by an updated normal vector  $\mathbf{n}_{2,0}$ .

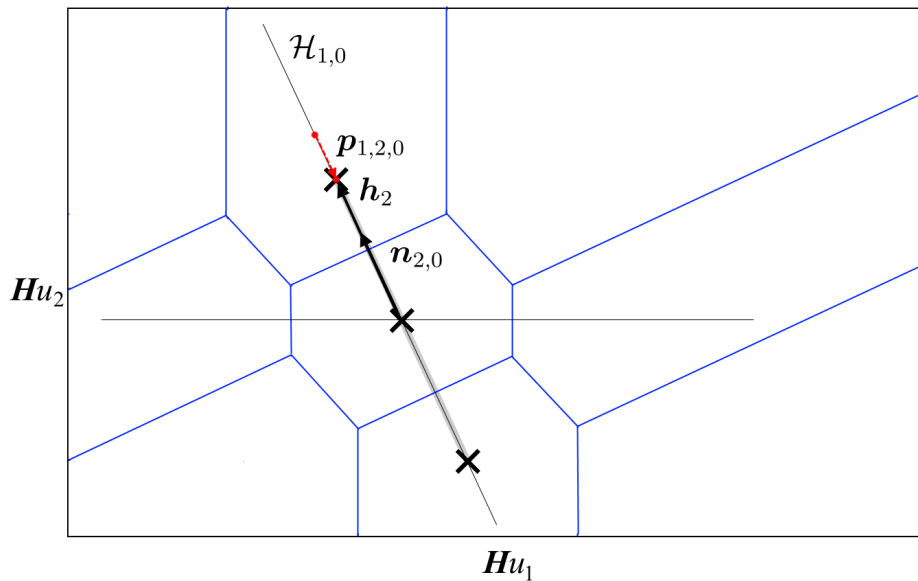


Figure 3.20: Orthogonal projection onto the  $(d-1)$ -dimensional parallelotope in the sub-space spanned by  $\mathcal{H}_{1,0}$ .

Minimum distance projection of the *translated* point  $(\mathbf{x} + \mathbf{p}_{1,1} - \mathbf{h}_1)$  onto the supporting hyperplanes results in the projection vector  $\mathbf{p}_{1,2,0}$ . The projection ending on a face of the  $(d-1)$ -parallelotope signifies that the convex hull has been reached, and that the projection vector  $\hat{\mathbf{p}}$  can be computed by summing the orthogonal projections, i.e.  $\hat{\mathbf{p}} = \mathbf{p}_{1,1} + \mathbf{p}_{1,2,0}$ . In summary, the proposed concept can be considered as finding the projection vector with minimum distance to the hull of a parallelotope, through the summation of a sequence of orthogonal projections which maximise the distance to the convex hull in their respective affine spaces.

### 3.6.3 Projection algorithm

In this section the computational details are presented in conjunction with the algorithm pseudo code, Algorithm 4. The algorithm requires as inputs the target  $\mathbf{x} \in \mathbb{R}^d$ , the

---

**Algorithm 4** *Projection algorithm*


---

```

1: function PROJH( $\mathbf{x}, \mathbf{H}, \mathbf{N}$ )
2:    $\tilde{\mathbf{x}} \leftarrow \mathbf{x}$ 
3:    $\hat{\mathbf{p}} \leftarrow \mathbf{0}$ 
4:    $\hat{k} \leftarrow \infty$ 
5:   while  $|\hat{k}| > 1$  do
6:      $\mathbf{k} = \mathbf{N}^T \mathbf{x}$ 
7:      $j^* = \arg \max_j ( (|\mathbf{k}_j| - 1) / \|\mathbf{n}_j\| )$  for all  $j = 1, 2, \dots, d$ 
8:      $\hat{k} = k_{j^*}$ 
9:      $s = \text{sign}(\hat{k})$ 
10:     $\hat{\mathbf{n}} = s \mathbf{n}_{j^*}$ 
11:     $\hat{\mathbf{h}} = s \mathbf{h}_{j^*}$ 
12:    if  $|\hat{k}| > 1$  then
13:       $\mathbf{p}^\perp = -((|\hat{k}| - 1) / \langle \hat{\mathbf{n}} \cdot \hat{\mathbf{n}} \rangle) \hat{\mathbf{n}}$ 
14:       $\mathbf{x} = \mathbf{x} + \mathbf{p}^\perp - \hat{\mathbf{h}}$ 
15:       $\mathbf{H}_{(:,j^*)} = \mathbf{0}$ 
16:       $\mathbf{N} = \mathbf{N} - \hat{\mathbf{n}} ((\mathbf{N}^T \hat{\mathbf{n}}) / \langle \hat{\mathbf{n}} \cdot \hat{\mathbf{n}} \rangle)^T$ 
17:       $\hat{\mathbf{p}} = \hat{\mathbf{p}} + \mathbf{p}^\perp$ 
18:    end if
19:  end while
20:  return  $\tilde{\mathbf{x}} = \tilde{\mathbf{x}} + \hat{\mathbf{p}}$ 
21: end function

```

---

transformation matrix  $\mathbf{H} \in \mathbb{R}^{d \times d}$  and the normal vectors  $\mathbf{N} \in \mathbb{R}^{d \times d}$  that define the lattice convex hull. Initial steps (lines:2-4), involve saving of the target, setting the minimum distance projection vector  $\hat{\mathbf{p}}$  to zero and the projection coefficient  $\hat{k}$  to infinity. The while-loop will repeat until the containment test (3.29) is satisfied as the result of an orthogonal projection that ends on a *face* of the convex hull. As a first step (line:6), the target in the *current subspace* is projected onto the set of normal vectors to obtain a vector of projection coefficients  $\mathbf{k}$  (3.28). Note that due to the symmetry of the convex hull, the hull hyperplane pairs are reflections in the origin, and only one hyperplane per normal vector will be relevant when projecting from a point *outside* onto the convex hull. The signs of the individual projection coefficients  $k_j$  indicate which hull hyperplanes of the  $\pm$  pairs are of relevance. Computation of the individual projection vector magnitudes

(line:7) is derived from (3.22) and (3.24) to give

$$\|\mathbf{p}_j\| = (|k_{j,x}| - k_{j,h}) \|\mathbf{n}_j\| \quad (3.62a)$$

$$= \frac{|\langle \mathbf{n}_j \cdot \mathbf{x} \rangle| - 1}{\langle \mathbf{n}_j \cdot \mathbf{n}_j \rangle} \|\mathbf{n}_j\| \quad (3.62b)$$

$$= \frac{|k_j| - 1}{\|\mathbf{n}_j\|} \quad (3.62c)$$

where  $k_j \in \mathbf{k}$ . Maximising the projection magnitudes (line:7) returns the index  $j^*$  which allows for identifying the corresponding projection coefficient and its relative sign ( $s = \pm 1$ ). From this, the normal  $\hat{\mathbf{n}}$  to, and translation  $\hat{\mathbf{h}}$  of the optimal hull hyperplane are identified. (Line:12) again applies the containment test and verifies if this latest projection ended on the face of the convex hull contained in the optimal hull hyperplane. If not, the maximum distance projection vector  $\mathbf{p}^\perp$  is computed (line:13). Note that  $\mathbf{p}^\perp$  is in an opposite direction (inward) to  $\hat{\mathbf{n}}$ . The next step updates the target with  $\mathbf{p}^\perp$ , so that it resides in the affine space spanned by the optimal hull hyperplane. According to the latest hull test, the point  $(\mathbf{x} + \mathbf{p}^\perp)$  is not contained in the face of the parallelotope hull, and thus minimum distance projection onto the relevant hull face is now required. To facilitate the process, the translation  $\hat{\mathbf{h}}$  of the optimal hull hyperplane is removed (line:15) to constitute a subspace defined by the updated basis  $\mathbf{H}$ . Subtracting  $\hat{\mathbf{h}}$  (line:14) from  $(\mathbf{x} + \mathbf{p}^\perp)$  moves it accordingly and updates the target in the new subspace.

In this new subspace, the hull face is a parallelotope of reduced dimension and requires an updated set of normal vectors. The easiest method is to replace the identified basis vector  $\mathbf{h}_j$  in the transformation matrix  $\mathbf{H}$  with  $\hat{\mathbf{n}}$ , and to recalculate the transposed inverse thereof  $\mathbf{N} = \mathbf{H}^{-T}$ . This approach requires the matrix inverse of  $\mathbf{H}$ , which we prefer to avoid for its computational requirement. As another option we orthogonalize the normals in the current set  $\mathbf{N}$  to the identified normal vector  $\hat{\mathbf{n}}$ . This process is similar to the first step in the *Gram-Schmidt orthogonalisation*<sup>14</sup>, whereby vectors in a set are projected onto the null-space of a specific vector. The result is a set of vectors which are orthogonalised to the specific vector, while their relation to one another remains intact.

<sup>14</sup>The Gram-Schmidt orthogonalisation procedure takes any set of  $d$  linearly independent vectors and modifies it to produce a set of  $d$  orthonormal vectors [110]. Fundamentally, it projects each vector onto the space orthogonal to the previous vectors.

Computation of the new set of normals equates to

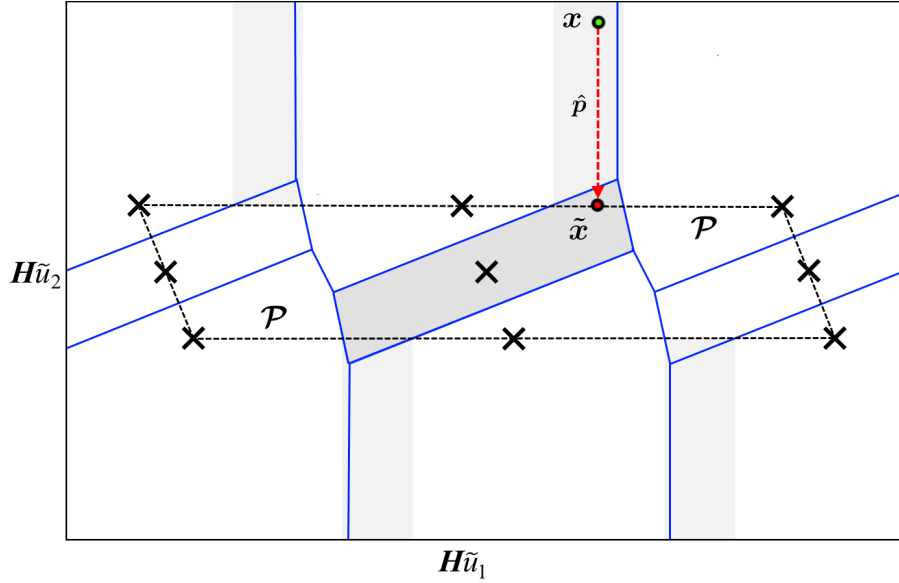
$$\mathbf{n}_j = \mathbf{n}_j - \frac{\langle \hat{\mathbf{n}} \cdot \mathbf{n}_j \rangle}{\langle \hat{\mathbf{n}} \cdot \hat{\mathbf{n}} \rangle} \hat{\mathbf{n}} \quad \text{for all } j = 1, 2, \dots, d. \quad (3.63)$$

Note that the normal vector  $\hat{\mathbf{n}}$  in the updated set  $\mathbf{N}$  nullifies. The last line in the while-loop composes the projection vector  $\hat{\mathbf{p}}$  by including the current orthogonal projection  $\mathbf{p}^\perp$ . While the hull test remains negative, the process repeats and finds the maximum orthogonal projections in the optimal hull hyperplanes. Note that with every iteration of the while-loop the parallelotope is reduced in dimension until an orthogonal projection ends on a relevant hull face. The projection vector  $\hat{\mathbf{p}}$  is also continuously updated with the orthogonal projections  $\mathbf{p}^\perp$  until the process stops. As a last step the point on the  $d$ -dimensional convex hull with minimum distance to the original target is computed by adding the projection vector to the original target (line:20).

### 3.6.4 Optimality

Calling on the projection algorithm to assist in solving the CVP (3.32) results in some occasional inaccuracies. This loss in optimality is similar to the issues reported by [106], where the errors incurred were attributed to the conditioning of the lattice, typical to prediction horizons above  $N = 5$  and specific to the case study presented. In [95], the irregularities were also noted, but the influence thereof on the control problem was deemed negligible.

The research presented in this writing focused on the decoding strategy of the MVA, which is based on the basic Voronoi cell of a lattice and recursive tiling of the lattice search space therewith (Fig. 3.16). This necessitated a heightened interest in the decoding inaccuracies as a result of the projection process. Upon investigation of the solution space for the three-phase NPC, it was noted that the errors occur in instances where the *orthogonality defect* (3.11) of the lattice basis is relatively high. In such cases, the partitioning of the solution space often results in the basic Voronoi cell of the lattice extending beyond the outer boundary of the lattice convex hull. Figure 3.1a exemplifies the issue, showing a lattice with increased orthogonality defect and basic Voronoi cell  $\mathcal{V}(\mathcal{L}, \mathbf{0}) \not\subseteq \mathcal{P}$ . These protrusions which extend beyond the convex hull of the lattice allow for some projections to end up inside a Voronoi region other than the one it originated from. The areas prone to such errors are lightly shaded in Figure 3.21.

Figure 3.21: Voronoi diagram with  $\mathcal{V}(\mathcal{L}, \underline{\mathbf{0}}) \not\subseteq \mathcal{P}$ .

### 3.6.4.1 Projection hull

In an attempt to address the problem, Raath(2017) [103] proposed that the projection must be carried out onto an enlarged *projection hull*  $\mathcal{P}^*$ . Defining an enlarged hull to project onto aims at containing the basic Voronoi cell  $\mathcal{V}(\mathcal{L}, \underline{\mathbf{0}}) \subseteq \mathcal{P}^*$ , which will prevent sub-optimal projections. The ideal would be to define an *enlarged* version of the lattice hull that will bound  $\mathcal{V}(\mathcal{L}, \underline{\mathbf{0}})$  as closely as possible. This will ensure that  $\tilde{\mathbf{x}}$  is in close proximity to  $\mathcal{P}$  and contained in a tiling of the Voronoi cell centered around the lattice points in the outer bound. Obtaining such an exact bound requires the vertices of the basic Voronoi cell; unfortunately a process that has proven to be NP-hard in high dimensions [111]. Raath(2017) [103] proposed to enlarge the projection hull in such a manner that the supporting hull hyperplanes with minimum distance to the origin is at least equal to the *covering radius* of the lattice  $\mu(\mathcal{L})$ . This effects containment of the covering sphere and subsequently the basic Voronoi cell  $\mathcal{V}(\mathcal{L}, \underline{\mathbf{0}})$ . Computing  $\mu(\mathcal{L})$  exactly is also NP-hard [112], therefore the *approximated covering radius* of a lattice is considered [46]

$$\rho = \frac{1}{2} \sqrt{\sum_{j=1}^d \|\mathbf{h}_j^*\|^2} \geq \mu(\mathcal{L}). \quad (3.64)$$

The vectors  $\mathbf{h}_j^*$  denote the *Gram Schmidt orthogonalisation* of the lattice basis vectors  $\mathbf{h}_j \in \mathbf{H}$ . Definition of the enlarged projection hull equates to moving the lattice hull

hyperplanes outwards in the direction of their respective normals with a distance of

$$\partial\rho = \rho - \min \frac{1}{\|\mathbf{n}_j\|} \quad \text{for } j = 1, 2, \dots, d. \quad (3.65)$$

We recall that  $1/\|\mathbf{n}_j\|$  (3.22) represents the orthogonal distance of the respective lattice hull hyperplanes from the origin. Accordingly, the new distances from the origin are

$$\frac{1}{\|\mathbf{n}_j^*\|} = \frac{1}{\|\mathbf{n}_j\|} + \partial\rho = (1 + \partial\rho \|\mathbf{n}_j\|) \frac{1}{\|\mathbf{n}_j\|}, \quad (3.66)$$

which allows for defining the set of normals  $\mathbf{n}_j^* \in \mathbf{N}^*$  by

$$\mathbf{n}_j^* = \frac{1}{(1 + \partial\rho \|\mathbf{n}_j\|)} \mathbf{n}_j, \quad \text{for } j = 1, 2, \dots, d. \quad (3.67)$$

Similar to (3.27), the enlarged projection hull is defined as

$$\mathcal{P}^*(\mathcal{L}(\mathbf{H})) = \{\mathbf{x} \in \mathbb{R}^d : |\langle \mathbf{n}_j^*, \mathbf{x} \rangle| \leq 1, \quad \text{for } j = 1, 2, \dots, d\}. \quad (3.68)$$

Figure 3.22 shows the enlarged *projection hull*  $\mathcal{P}^*$  with the integrity of the updated target  $\mathbf{x}^*$  after orthogonal projection intact.

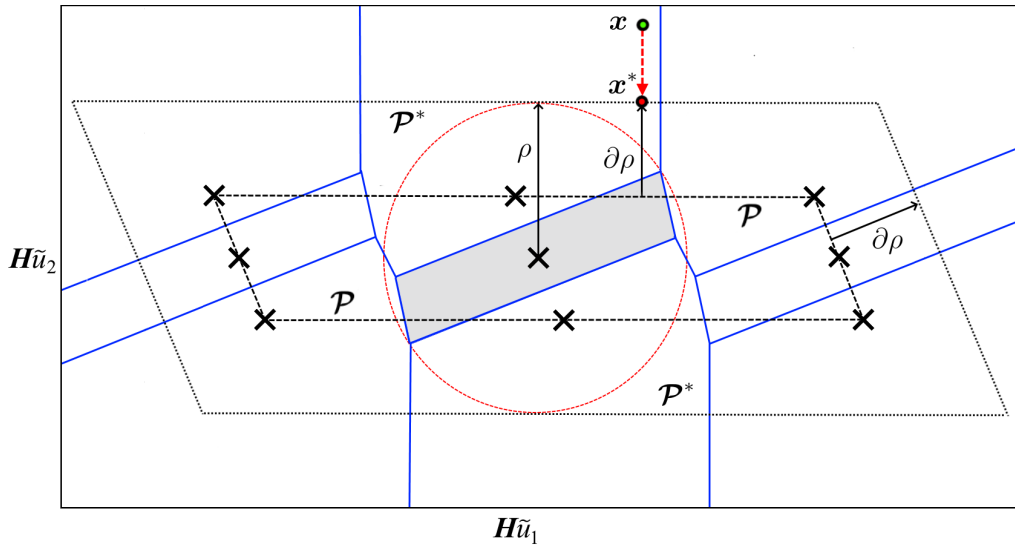


Figure 3.22: Projection onto enlarged convex hull  $\mathcal{P}^*$ .

Incorporating this approach in the projection algorithm, calls for a set of normals  $\mathbf{N}^*$  and a corresponding set of basis vectors  $\mathbf{h}_j^* \in \mathbf{H}^* = (\mathbf{N}^*)^{-T}$  which have to be computed offline and stored for online use. To eliminate this demand it is proposed to compute the approximated covering radius offline and avail only the partial difference  $\partial\rho$  for online use. The projection process is maintained as before where the updated target  $\tilde{\mathbf{x}}$  results from orthogonal projection  $\hat{\mathbf{p}}$  onto the lattice convex hull  $\mathcal{P}$ . Once  $\tilde{\mathbf{x}}$  is found the orthogonal

projection  $\hat{\mathbf{p}}$  is reduced with  $\partial\rho$  to resemble projection onto an enlarged projection hull with updated target

$$\mathbf{x}^* = \tilde{\mathbf{x}} - \left( \frac{\|\hat{\mathbf{p}}\| - \partial\rho}{\|\hat{\mathbf{p}}\|} \right) \hat{\mathbf{p}} \quad (3.69a)$$

$$= \tilde{\mathbf{x}} - \left( 1 - \frac{\partial\rho}{\|\hat{\mathbf{p}}\|} \right) \hat{\mathbf{p}} \quad (3.69b)$$

Figure 3.23 displays the principle geometrically for the two-dimensional case. Although the approach achieves the objective, the illustrative example shows that the basic Voronoi cell is still not optimally bounded by  $\mathcal{P}^*$ . If an optimal bound is required, algorithms such as the diamond cutting algorithm [113] can be considered for computing the vertices of the Voronoi cell and the subsequent hull enlargement.

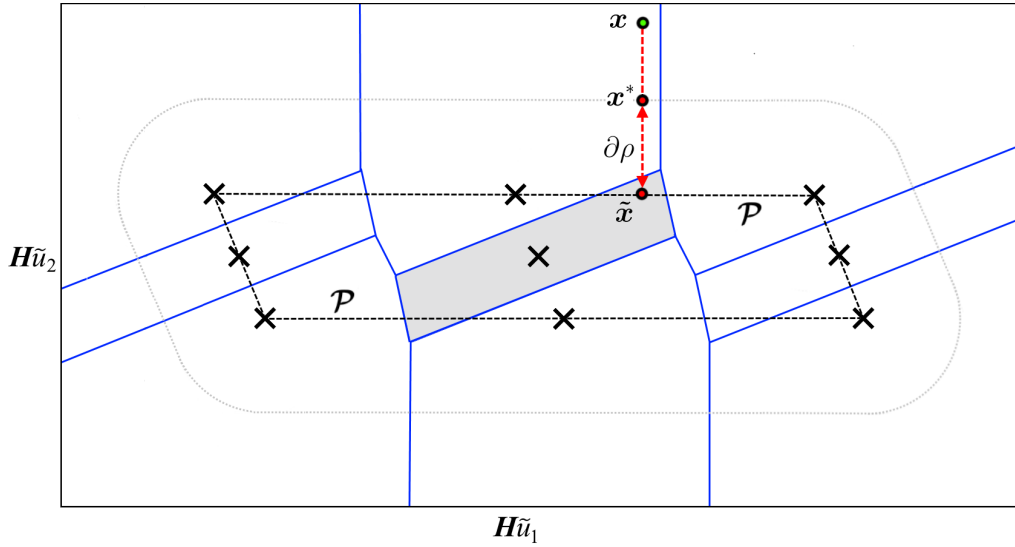


Figure 3.23: Backtracking on the projection trajectory, simulating an enlarged projection hull.

### 3.6.5 Projection algorithm with enlargement factor

Incorporating the proposed approach into Algorithm 4 requires the enlargement factor  $\partial\rho$  as an additional input. The projection process commences as before, and computes the minimum distance projection onto the lattice convex hull. Once the updated target  $\tilde{\mathbf{x}}$  is obtained, the enlargement factor is considered to determine whether the covering radius exceeds the distance to any of the lattice hull hyperplanes ( $\partial\rho > 0$ ). If the inequality is satisfied, sub-optimality may exist, and further consideration of the projection vector  $\hat{\mathbf{p}}$  is required. The projection vector is compared to the enlargement factor, and is only adjusted if  $\|\hat{\mathbf{p}}\| > \partial\rho$ . Algorithm 5 lists the altered pseudo-code (lines:1 and 21 to 25).

**Algorithm 5** *Projection algorithm*


---

```

1: function PROJHOPT( $\mathbf{x}, \mathbf{H}, \mathbf{N}, \partial\rho$ )
2:    $\tilde{\mathbf{x}} \leftarrow \mathbf{x}$ 
3:    $\hat{\mathbf{p}} \leftarrow \mathbf{0}$ 
4:    $\hat{k} \leftarrow \infty$ 
5:   while  $|\hat{k}| > 1$  do
6:      $\mathbf{k} = \mathbf{N}^T \mathbf{x}$ 
7:      $j^* = \arg \max_j ( (|k_j| - 1) / \|\mathbf{n}_j\| )$  for all  $j = 1, 2, \dots, d$ 
8:      $\hat{k} = k_{j^*}$ 
9:      $s = \text{sign}(\hat{k})$ 
10:     $\hat{\mathbf{n}} = s \mathbf{n}_{j^*}$ 
11:     $\hat{\mathbf{h}} = s \mathbf{h}_{j^*}$ 
12:    if  $|\hat{k}| > 1$  then
13:       $\mathbf{p}^\perp = -((|\hat{k}| - 1) / \langle \hat{\mathbf{n}} \cdot \hat{\mathbf{n}} \rangle) \hat{\mathbf{n}}$ 
14:       $\mathbf{x} = \mathbf{x} + \mathbf{p}^\perp - \hat{\mathbf{h}}$ 
15:       $\mathbf{H}_{(:,j^*)} = \mathbf{0}$ 
16:       $\mathbf{N} = \mathbf{N} - \hat{\mathbf{n}} ((\mathbf{N}^T \hat{\mathbf{n}}) / \langle \hat{\mathbf{n}} \cdot \hat{\mathbf{n}} \rangle)^T$ 
17:       $\hat{\mathbf{p}} = \hat{\mathbf{p}} + \mathbf{p}^\perp$ 
18:    end if
19:  end while
20:   $\tilde{\mathbf{x}} = \tilde{\mathbf{x}} + \hat{\mathbf{p}}$ 
21:   $\mathbf{x}^* = \tilde{\mathbf{x}}$ 
22:  if  $\partial\rho > 0$  then
23:    if  $\|\hat{\mathbf{p}}\| > \partial\rho$  then
24:       $\mathbf{x}^* = \tilde{\mathbf{x}} - (1 - \partial\rho / \|\hat{\mathbf{p}}\|) \hat{\mathbf{p}}$ 
25:    end if
26:  end if
27:  return  $\mathbf{x}^*$ 
28: end function

```

---

**3.6.5.1 Complexity**

Analysis of Algorithm 5 reveals that (lines: 6,7 and 16) demand the most computations. One full *iteration* of the algorithm requires a minimum of

$$(9d^2 + 10d + 2) \text{ flops} \quad (3.70)$$

which exclude the final summation (line:20) and optimality verification (lines:21-25) totaling  $(7d - 1)$  flops. The number of iterations required to solve the projection problem is influenced by the geometrical position of the target relative to the lattice convex hull. This dictates on which dimensional face the minimum distance projection will end. In a worst case scenario where the nearest point on the convex hull to the target is a vertex (zero-dimensional face), the algorithm is compelled to step through all dimensions



$d, (d-1), \dots, (1)$ . Hence,  $d$ -iterations which bound the algorithm's computational cost to

$$(9d^3 + 10d^2 + 9d - 1) \text{ flops.} \quad (3.71)$$

### 3.7 Summary

In this chapter the FCS-MPC problem were demonstrated as the CVP of a truncated lattice. Various solvers to the problem was presented together with their respective algorithms and accompanying complexities. The SDA posed the lowest computational complexity in terms of flop count while the ISA and MVA exhibited their deterministic natures. Solving the CVP in deterministic single exponential time is an exciting prospect, however, the exponential space requirement signals caution. For all the solvers an initial *estimated* solution is favoured to speed up the respective optimisation processes. The issue of a distant target to a constrained search space was also presented, and as counter, the principle of orthogonal projection onto the lattice convex hull. A projection algorithm was proposed which solves the problem and also considers the optimality issue that occasionally arises with this preconditioning approach.

## Chapter 4

# Performance estimation and development

In this chapter, the general performance of the MVA as solver to the truncated CVP, unique to FCS-MPC of a power electronic system (plant) is presented. Attention is focused on the decoding algorithm's operation during steady-state and transient conditions of the system. The power electronic system selected for control is a three-phase, three-level, diode-clamped multilevel inverter driving a passive  $RL$ -load. Such a first-order system with fast dynamic response simplifies the MPC formulation and easy the control process. The reason for this selection is that the performance of the solvers is of main concern and not the MPC process itself. As a benchmark, the well-acclaimed SDA of [39] was opted for, with an initial sphere radius selection strategy which involves the so-called Babai estimate (3.48). The projection algorithm presented in the previous chapter is employed to precondition the CVP target and assist the respective decoders. The chapter starts off with the modeling of the power electronic system and formulation of the MPC process. Simulation results follow with the performance analysis and observations concluding the chapter.

### 4.1 Inverter with $RL$ -load

Consider the layout of the FCS-MPC structure demonstrated in Figure 4.1 featuring a power electronic system (plant) consisting of a three-phase, NPC inverter driving a balanced  $RL$ -load. The circuits and variables for the inverter and  $RL$ -load blocks are adopted

from sub-sections 2.5.1 and 2.5.2 respectively. We recall that the NPC is commanded by

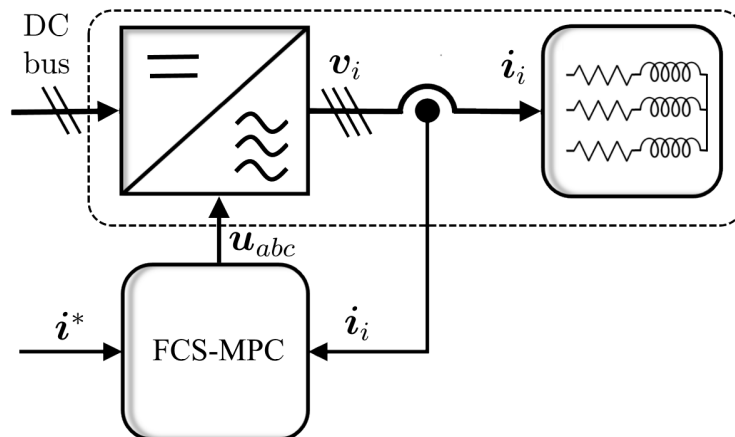


Figure 4.1: Three-phase 3L NPC VSC with  $RL$ -load.

a three-dimensional control vector  $\mathbf{u}_{abc}$  (2.45) where each element ( $u_x$ ) determines the state of an individual inverter-leg. The set of leg-states for three voltage levels  $\mathcal{U}$  (2.42), effects the inverter control set  $\mathcal{U}$  (2.46) to pose 27 different control vectors.

**Control objective:** To control the state of the three inverter legs by means of  $\mathbf{u}_{abc}$ , so that the load current  $\mathbf{i}_i$  closely tracks its reference  $\mathbf{i}^*$  while limiting the average switching frequency and adhering to the inverter switching constraint (2.49).

**Assumptions:** Neutral point voltage of the NPC is regulated and constant at  $V_{dc}/2$ .

**FCS-MPC process:** The operational process of the controller follows the same pattern as the fundamental MPC formulation described in section 2.2.1. At every sampling instant ( $k$ ), the inverter current i.e. load current  $\mathbf{i}_i$  is measured. This current is used as the initial state when calling on the *predictive model* of the system. The future load current and the control sequence  $\tilde{\mathbf{u}}$  that effects it, are included in the *cost function* that defines the control system objectives. The control sequence that mimics the unconstrained minimum  $\tilde{\mathbf{u}}_{unc}$  in the closest manner, is selected as the optimal control sequence  $\hat{\mathbf{u}}$ . From this sequence, the first control vector is applied to the inverter

$$\mathbf{u}_{abc} = \hat{\mathbf{u}}_{1:3}. \quad (4.1)$$

### 4.1.1 System model

Modelling of the proposed system proceed as follows: the NPC produces a three-phase output voltage  $\mathbf{v}_i$  with three levels, as defined by the control set  $\mathcal{U}$ . Representation in the

stationary  $\alpha\beta$  coordinates gives

$$\mathbf{v}_i = [v_{i\alpha} \ v_{i\beta}]^T = \frac{V_{dc}}{2} \mathbf{K} \mathbf{u}_{abc}, \quad (4.2)$$

where  $\mathbf{K}$  denotes the reduced Clarke transformation (2.38). Applying the inverter output voltage to the  $RL$ -load induces the load current  $\mathbf{i}_i = [i_{i\alpha} \ i_{i\beta}]^T$ . For the control problem at hand, and with the load current selected as state variable, the continuous-time state-space equations of the system are

$$\frac{di_{i\alpha}(t)}{dt} = -\frac{R}{L}i_{i\alpha}(t) + \frac{v_{i\alpha}}{L} \quad (4.3a)$$

$$\frac{di_{i\beta}(t)}{dt} = -\frac{R}{L}i_{i\beta}(t) + \frac{v_{i\beta}}{L}. \quad (4.3b)$$

In matrix form the system model then becomes

$$\frac{d\mathbf{i}_i(t)}{dt} = \mathbf{F}\mathbf{i}_i(t) + \mathbf{G}\mathbf{u}_{abc}(t) \quad (4.4a)$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{i}_i(t) \quad (4.4b)$$

with state vector  $\mathbf{i}_i$ , input vector  $\mathbf{u}_{abc}$  and their respective matrices

$$\mathbf{F} = -\frac{R}{L} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

$$\mathbf{G} = \frac{V_{dc}}{3L} \begin{bmatrix} 1 & -1/2 & -1/2 \\ 0 & \sqrt{3}/2 & -\sqrt{3}/2 \end{bmatrix},$$

$$\mathbf{C} = \mathbf{I}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Converting the differential equations to the discrete time domain with sampling interval  $T_s$  results in the discrete-time model

$$\mathbf{i}_i(k+1) = \mathbf{A}\mathbf{i}_i(k) + \mathbf{B}\mathbf{u}_{abc}(k) \quad (4.5a)$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{i}_i(k), \quad (4.5b)$$

where

$$\mathbf{A} = e^{\mathbf{F}T_s} \quad \text{and} \quad \mathbf{B} = -\mathbf{F}^{-1}(\mathbf{I}_2 - \mathbf{A})\mathbf{G}.$$

### 4.1.2 MPC formulation

The general cost function for FCS-MPC with reference tracking (2.13) is adopted as optimisation criterion for the plant. Hence, the relevant optimisation problem (2.36) with congruent matrices are considered. Tracking of one state variable  $\mathbf{x} = \mathbf{i}_i$ , eliminates the need for selective penalties, i.e.  $\mathbf{\Lambda} = \mathbf{I}_2$ . The system parameters sets the respective matrix dimensions to  $\tilde{\mathbf{\Lambda}} \in \mathbb{N}^{2N \times 2N}$ ,  $\mathbf{\Gamma} \in \mathbb{R}^{2N \times 2}$ ,  $\mathbf{\Upsilon} \in \mathbb{R}^{2N \times 3N}$ ,  $\mathbf{S} \in \mathbb{N}^{3N \times 3N}$ ,  $\mathbf{E} \in \mathbb{N}^{3N \times 3}$ ,  $\mathbf{Q} \in \mathbb{R}^{3N \times 3N}$  and  $\mathbf{H} \in \mathbb{R}^{3N \times 3N}$ . Auxiliary matrices  $\mathbf{S}$  and  $\mathbf{E}$  are constructed from the diagonal matrices  $\mathbf{I}_3$  and  $\mathbf{0}_3$ . Dimensions of the control sequence (2.9) and output sequence (2.19) respectively are  $\tilde{\mathbf{u}}(k) \in \mathbb{Z}^{3N}$  and  $\tilde{\mathbf{y}}(k) \in \mathbb{R}^{2N}$ .

## 4.2 Simulation: performance evaluation

In an attempt to quantify the performance of the respective decoding algorithms, the power electronic system presented above was simulated in MATLAB<sup>®</sup>. The system parameters were selected to resemble a typical MV application as presented in [5]. A sampling interval of  $T_S = 25\mu\text{s}$ , load resistance of  $R = 2\Omega$ , and load inductance  $L = 2\text{mH}$  were chosen. Supplied from a dc-link voltage of  $V_{DC} = 5.2\text{kV}$ , the expected, or rated r.m.s. output voltage of the inverter is  $V_{AC} = 3.3\text{kV}$  at a fundamental frequency of 50Hz. Base quantities were used to establish a per-unit system with the reference current amplitude assumed to be 0.8pu. The tuning factor  $\lambda_u$  was adjusted to achieve, as per (2.51), an average switching frequency per semiconductor device of approximately  $f_{swd} = 300\text{Hz}$ . Simulation of the FCS-MPC process followed a general order as illustrated by the state-machine in Figure 4.2.

At every sample period, the output variable is obtained from which the unconstrained minimum  $\tilde{\mathbf{u}}_{unc}(k)$  is computed. Transformation to  $\mathbf{H}$ -coordinate space sets the CVP target

$$\boldsymbol{\xi}(k) = \mathbf{H}\tilde{\mathbf{u}}_{unc}(k).$$

If necessary, the projection algorithm conditions the target to give the updated target  $\boldsymbol{\xi}^*(k)$ . The next step computes the initial sphere radius from the Babai estimate (3.48). In the  $\mathbf{H}$ -coordinate space, the Babai estimate  $\mathbf{H}\tilde{\mathbf{u}}_{bab}$  yields the same result as the rounded least squares solution (3.54), which identifies the lattice point  $\mathbf{s}(k)$  for the MVA to start

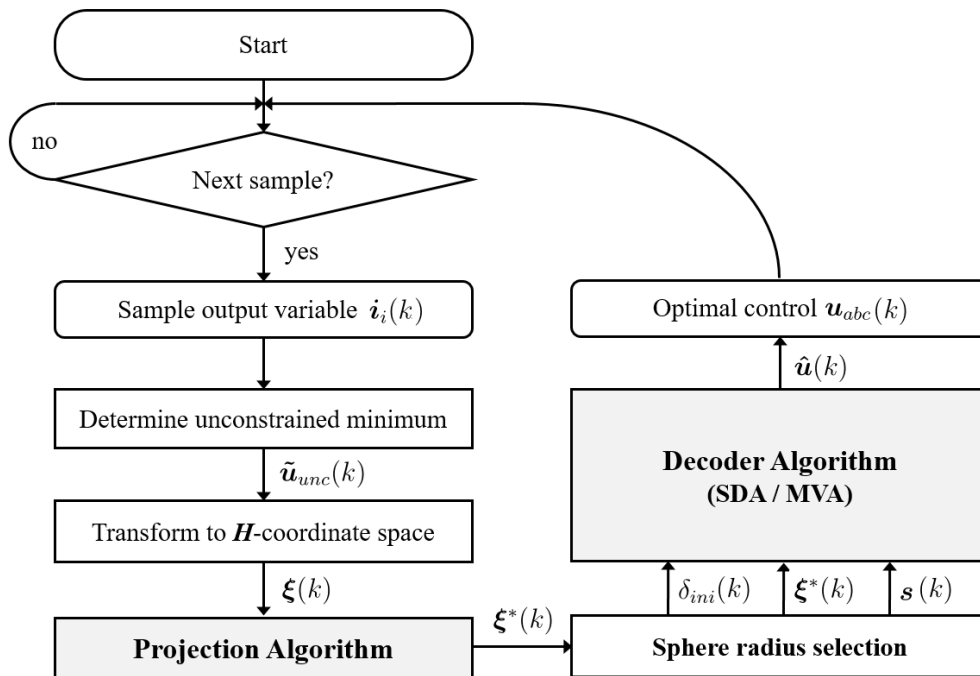


Figure 4.2: State machine representation of the MPC scheme with target preconditioning.

from. After decoding, the solution to the optimisation problem  $\hat{\mathbf{u}}(k)$  (2.36a) is produced, from which the first control vector  $\mathbf{u}_{abc}(k)$  is extracted and applied as the optimal control action. To obtain a comprehensive indication of the MVA's performance, it is necessary to consider two special operational conditions of the power electronic system, namely steady-state and transient conditions. As reference, two versions of the SDA will be considered, one operating from the original target  $\boldsymbol{\xi}$  and one operating from a preconditioned target  $\boldsymbol{\xi}^*$ . The SDA *with preconditioned target* will be referred to as the SDA\*. Where the results refer to the MVA or SDA\* *combined with* the projection algorithm (PA), the respective combinations will be denoted by SDAp and MVAp. Table 4.1 summarises the mentioned denotations.

Table 4.1: Algorithm denotations for experimental results.

Abbreviation	Performing algorithm/s	Target preconditioning	Initial solution
SDA	Sphere Decoding algorithm	no	Babai
SDA*	Sphere Decoding algorithm	yes	Babai
MVA	Micciancio Voulgaris algorithm	yes	Babai
SDAp	(SDA* + PA) unit	yes	Babai
MVAp	(MVA + PA) unit	yes	Babai

### 4.2.1 Steady-state conditions

Steady state conditions were simulated with a system “hot-start”<sup>15</sup> and a current reference set at 0.8pu. The simulation was conducted for a period of 10 cycles of the fundamental output ac frequency. For the horizon  $N = 5$  case, Figure 4.3 illustrates over two fundamental periods (a) the inverter phase voltages and (b) the resulting three-phase load currents in pu-values. It can be seen that the load current waveforms accurately track their references (dotted lines).

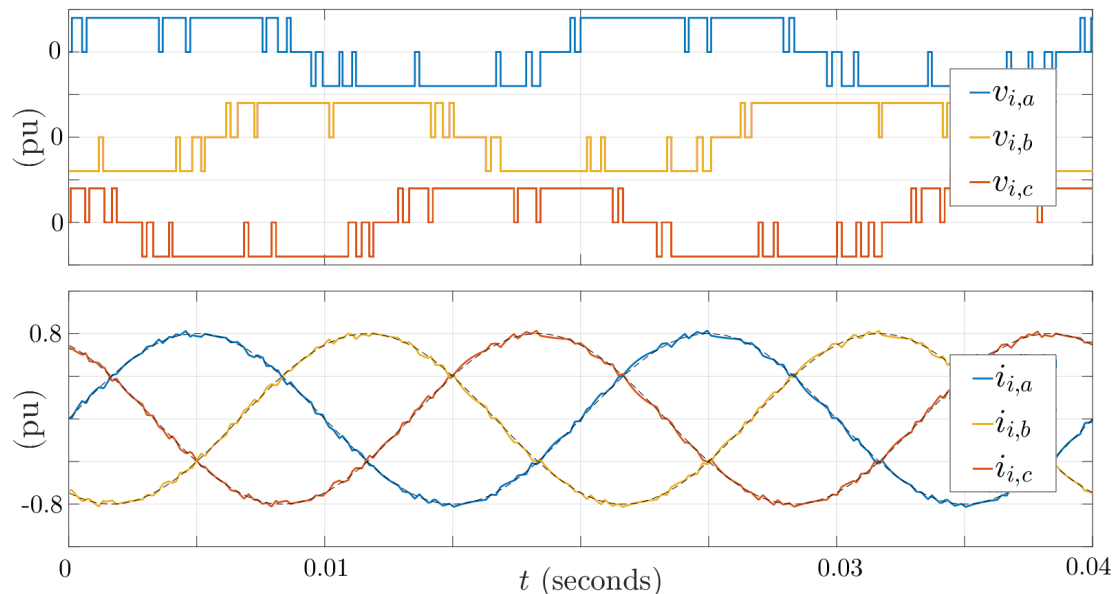


Figure 4.3: System waveforms for steady-state operation of the 3-level NPC inverter during steady-state conditions. (a) The inverter output voltage  $v_i$  and (b) the  $RL$ -load current  $i_i$ .

Recording the maximum iterations required by each individual algorithm resulted in the graph plotted in Figure 4.4 for prediction horizons 1 to 10. Almost similar efforts are exerted by the two SDAs; however, the SDA\* equipped with target preconditioning in some instances require more iterations than its counterpart. The MVA on the other hand, exhibits a low number of algorithm iterations, hinting at its deterministic nature. Closely tracking its upper bound (see subsection 3.6.5.1), the iterations of the projection algorithm remained below the dimension of the CVP.

<sup>15</sup>With a hot-start, the simulation is initialised (sampling instant  $k = 0$ ) with the output current or system state set equal to its reference.

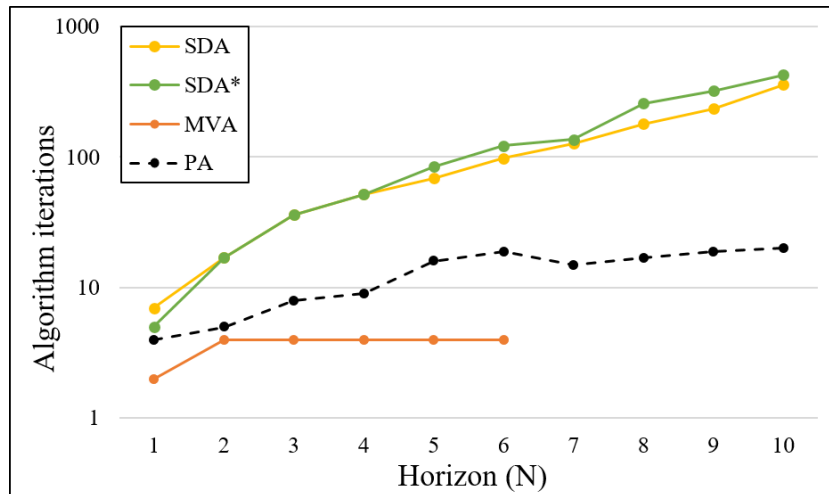


Figure 4.4: Maximum number of iterations required by the respective algorithms during steady-state conditions. The sphere decoding algorithm with a preconditioned target is denoted by SDA\*.

Converting the algorithm iterations to flop count reflects the computational burden required by each algorithm. The results per *individual* algorithm are displayed by the graphs in Figure 4.5a. In contrast to the small number of iterations required by the MVA, its flop count is relatively high. Congruent to their respective iterations, the maximum flop count of the two SDAs, exhibits similar characteristics with a small difference in total complexity. Note that the maximum flop count of the projection algorithm trends at or just below its theoretical upper bound (3.71). Since the MVA and SDA\* utilise the preconditioned target from the projection algorithm, the cost of preconditioning is added to the performance metric of the respective algorithms. Denoted as the MVAp and SDAp, the computational efforts, inclusive of the projection algorithm's burden, are showed in Figure 4.5b. This proves the conventional SDA without target preconditioning as the algorithm with least computational complexity in *steady-state* conditions.

Flop count being an indicator of algorithm complexity does not necessarily represent the algorithm's *computability*, i.e. the performance of an algorithm in terms of computation time [97]. In an attempt to quantify or at least compare the *relative* termination times of the different algorithms, the timing function of MATLAB<sup>®</sup> was called upon. Figure 4.6a plots the respective individual *maximum* algorithm termination times recorded. In Figure 4.6b, the termination time of the PA is added to the individual termination times of the MVA and SDA\*, resulting in the cumulative times denoted by MVAp and SDAp.



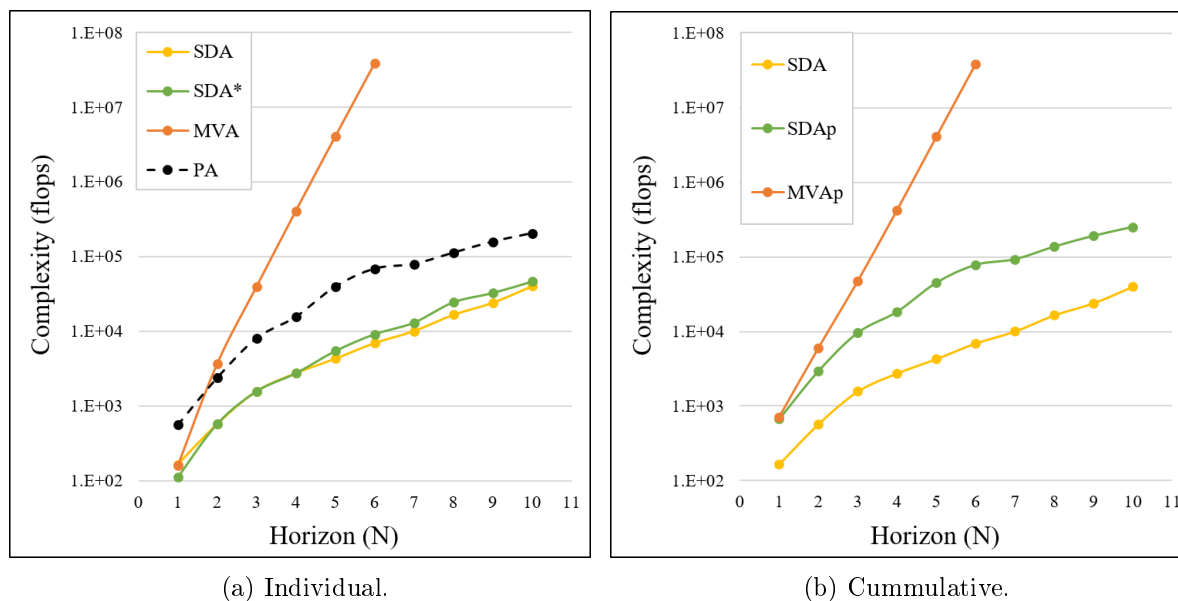


Figure 4.5: Maximum flops incurred during steady-state conditions. The sphere decoding algorithm with and without target preconditioning are respectively denoted by SDA\* and SDA. Reference to the performance of the MVA or SDA\* *combined with* the projection algorithm (PA), are respectively denoted by SDAp and MVAp.

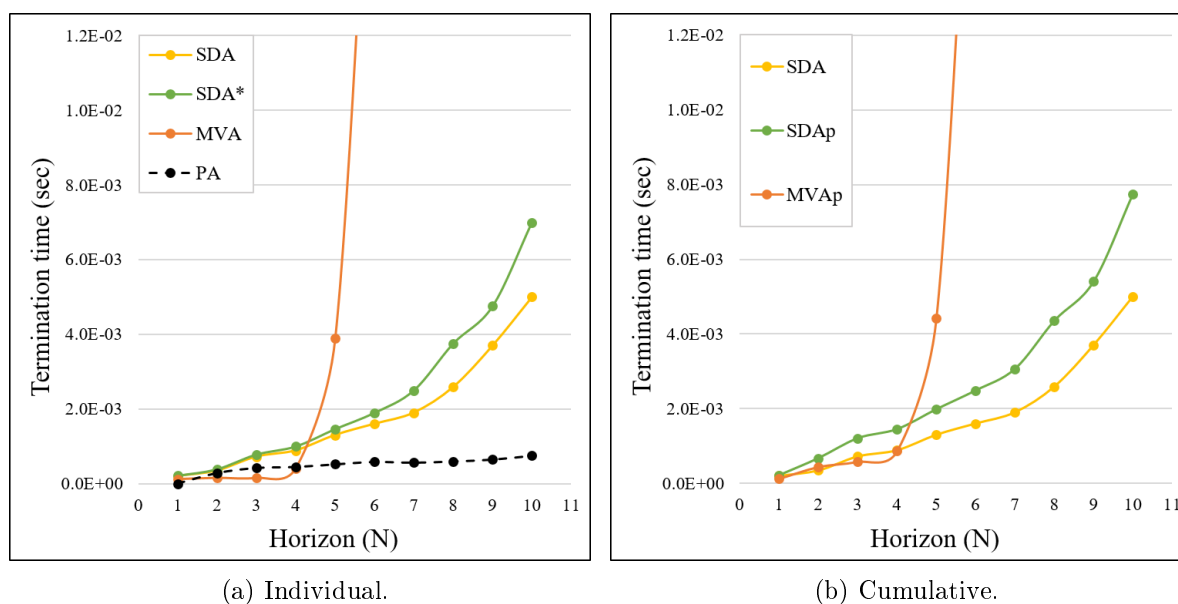


Figure 4.6: Maximum algorithm termination times during steady-state conditions. The sphere decoding algorithm with and without target preconditioning is respectively denoted by SDA\* and SDA. Reference to the performance of the MVA or SDA\* *combined with* the projection algorithm (PA), is respectively denoted by SDAp and MVAp.

Burdened with the highest flop count, the MVAp strategy recorded competitive termination times for prediction horizons 1 to 4. However, for horizons  $N > 5$  the computational burden of the MVA weighed down on the processing hardware, making it impractical to

consider for higher dimensions. Due to preconditioning of the target, the SDAP combination required more time to solve the CVP than the SDA without target preconditioning.

## 4.2.2 Transient conditions

To test the MVA's performance during transient conditions, the system is started from a zero-state with the reference current set to 0.8pu in amplitude. After a number of samples, another transient is introduced in the form of a step-down and step-up of the reference current, 0.8pu to 0.2pu and back up to 0.8pu. The resulting waveforms, again for the horizon  $N = 5$  case, are shown in Figure 4.7 over two periods of the fundamental frequency. In (b), the transient response of the MPC scheme can be observed as the load current

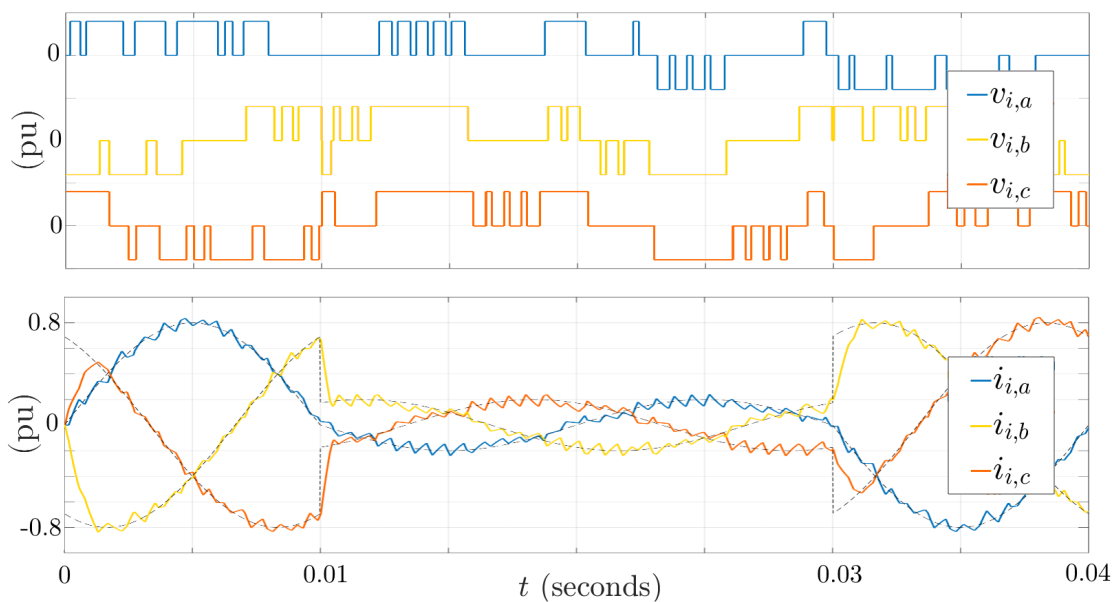


Figure 4.7: During transient conditions, (a) the inverter output voltage  $\mathbf{v}_{abc}$  and (b) the  $RL$ -load current  $\mathbf{i}_{abc}$ .

regains close proximity to the reference within milliseconds of the transient step. To enable effective tracking of the imposed transient steps, the optimal unconstrained solution typically resides outside and at a distance from the lattice convex hull. This complicates the CVP, which translates into an increased computational demand. The maximum iterations of the individual algorithms in Figure 4.8, shows that a dramatic increase is experienced by the conventional SDA. In comparison, the SDA\* with preconditioned target, recorded a moderate increase, whereas the MVA approximately remained the same as during steady-state conditions. As before, the projection algorithm tracked its upper

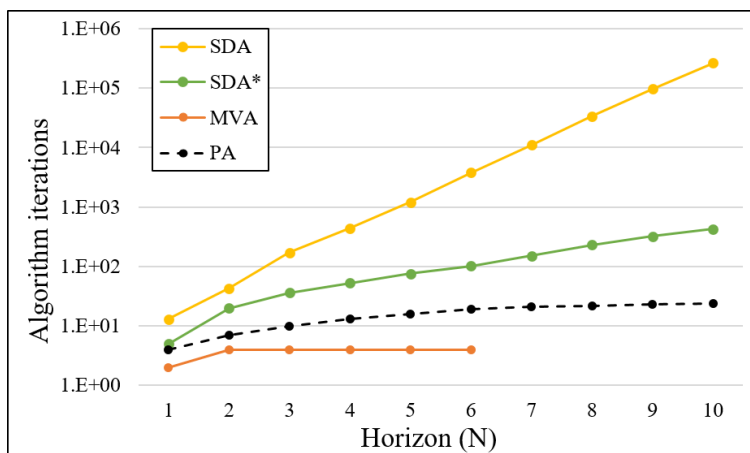


Figure 4.8: Maximum number of iterations required by the respective algorithms during transient conditions.

bound. Converting the maximum algorithm iterations to their corresponding flop counts (Fig. 4.9a), firstly, reiterates the deterministic nature of the MVA and secondly, exposes the vulnerability of the conventional SDA in transient conditions. It can also be concluded that the SDA\* has a much easier CVP to solve than the conventional SDA. Compared to steady-state conditions, the moderate increase in combined complexity (Fig. 4.9b) of the MVAp and SDAP strategies, substantiates the necessity of target preconditioning and the investment made in the projection algorithm.

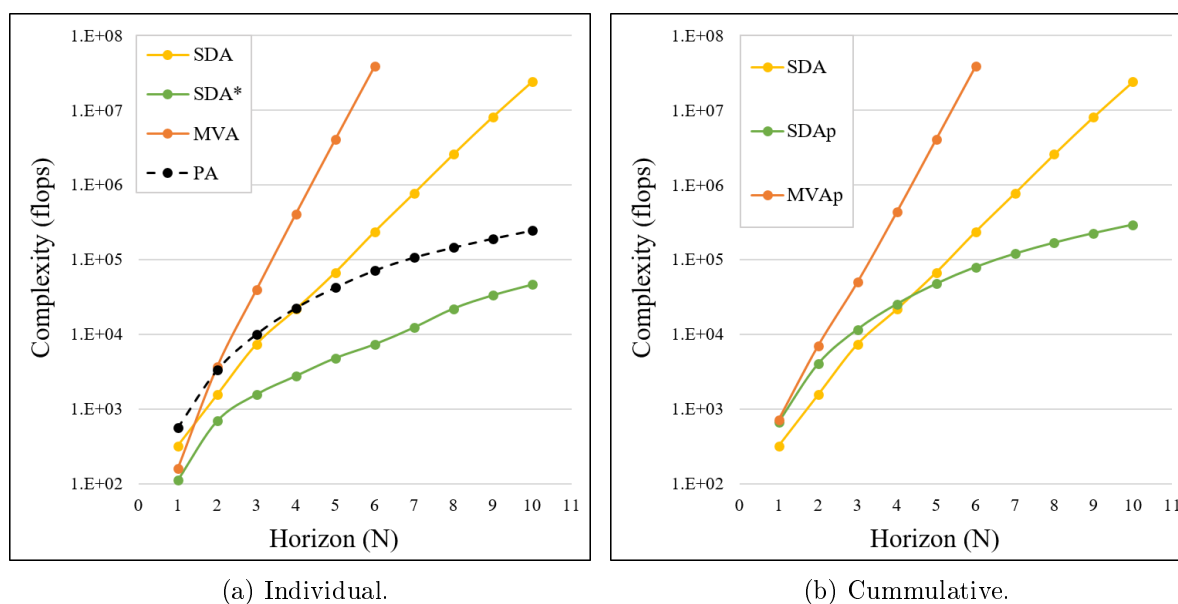


Figure 4.9: Maximum flops incurred during transient conditions.

The relative increase in algorithm complexities from steady- to transient state is also conveyed in the individual algorithm termination times (Fig. 4.10a). Clearly the SDA

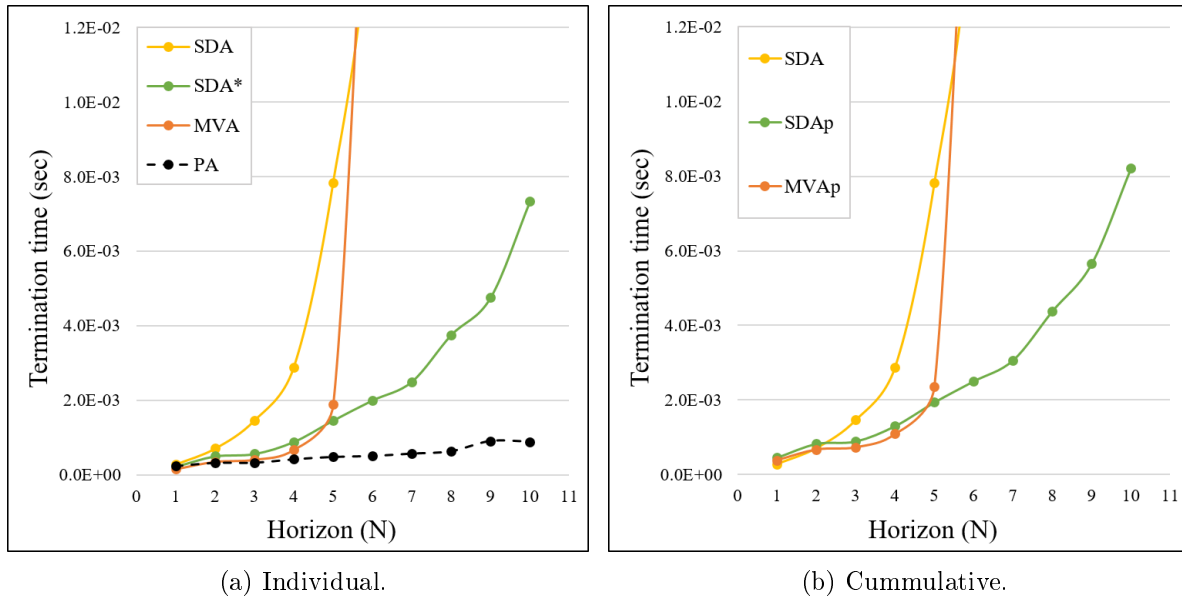


Figure 4.10: Average algorithm termination times per sample during transient conditions.

without target preconditioning is affected severely. The combined termination times (Fig. 4.10b) concludes that the MVA with its heightened complexity, terminates the fastest for horizons  $N = 1 - 4$ . However, for longer prediction horizons  $N \geq 5$ , the termination time of the MVA deteriorates drastically and is outperformed by the SDA\* equipped with target preconditioning.

#### 4.2.2.1 Cost of target preconditioning

Target preconditioning eases the CVP, but it requires some computational investment. In an attempt to evaluate or at least compare the projection algorithm proposed in this work, it was benchmarked against the respective optimisation functions of MATLAB<sup>®</sup> and the well known Multi-parametric toolbox (MPT)<sup>16</sup>. For the MATLAB<sup>®</sup> function “quadprog”, the projection problem is defined as a *box-constrained* quadratic program (QP) in the original or input space. In contrast, the MPT “distance” function solves the problem in the transformed  $\mathbf{H}$ -coordinate space. It computes the distance of a polyhedron from a point and also returns the location of the the point on the polyhedron.

Computability of the respective algorithms were quantified in terms of their individual

<sup>16</sup>“MPT is a software tool for MATLAB<sup>®</sup> that allows efficient formulation and solutions of optimisation problems involved in multi-parametric programming and computational geometry.” [114]

termination times, recorded with the MATLAB<sup>®</sup> timing function. The system simulations presented above were repeated under the same conditions (steady and transient) for prediction horizons 1 to 10. Figure 4.11 displays the *average termination times* required by the respective algorithms to solve the problem of minimum distance projection onto the *lattice convex hull*.

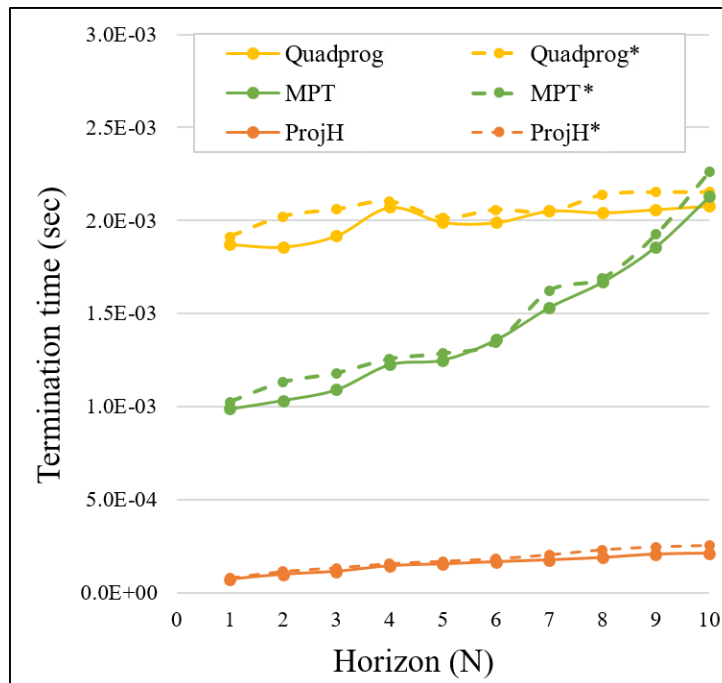


Figure 4.11: Average termination times of the MATLAB<sup>®</sup> “quadprog”, Multi-parametric toolbox (MPT) “distance” and Projection algorithm (ProjH) solvers for the minimum distance projection problem in steady-state and transient conditions(\*). Steady-state results are indicated with solid lines and transient conditions with dashed lines.

Comparatively, the proposed projection algorithm’s termination times compare favourably with those of the other solvers. Termination times of the MATLAB<sup>®</sup> optimiser “quadprog”, remained relatively constant throughout all the simulations, whereas the MPT “distance” function gradually increased with the dimension of the optimisation problem. In terms of accuracy, the projection algorithm and MPT function correlated with a maximum deviation in the norm of the solution to a margin of  $10^{-15}$ . The “quadprog” function in some cases deviated more as the default optimality tolerance for the “interior-point-convex” option was set at  $10^{-8}$ .

### 4.3 Summary

During steady-state operation of the plant, the target to the CVP remains in or relatively close to the convex hull of the truncated lattice. In terms of computational complexity, this situation favours the SDA without target preconditioning. Use of the projection algorithm aids the MVA to ensure optimality, but offers no advantage in either total complexity or computability to the SDA. Computability of the MVA during steady-state operations proved to be optimal for lower prediction horizons, but is surpassed by both SDA approaches from horizon 5. With the plant subjected to transient conditions, the CVP target is often located at a distance to the search space. In these occurrences, the projection algorithm intervenes and lowers the CVP's complexity. This added advantage allows for the SDA\* to pose the lowest individual complexity which, in combination with the projection algorithm (SDAp), exhibits the least deterioration in performance for CVPs of a higher dimension. Complexity and computability of the MVA remained relatively the same during steady-state and transient conditions. Noteworthy, while the MVA exhibited the highest complexity of all algorithms, it recorded low termination times for prediction horizons  $N \leq 4$ . This unexpected result raised suspicion and motivated further consideration. Upon investigation, the termination times of the MVA are supported by the following conclusions:

- The MVA exploits the recursive nature of a lattice, hence the low number of algorithm iterations required to solve the CVP. A lower number of iterations translate into fewer memory calls, which in turn effects the data transfer from memory to the CPU. Although a large block of data is moved in the MVA, it occurs once per iteration and on average takes four iterations to solve the CVP. In comparison, the SDA accesses a small amount of memory three times per iteration.
- In terms of computational resources, the MVA calls on matrix calculations, which engage the parallel processing capabilities of the CPU<sup>16</sup>. The sequential nature, i.e. low complexity of the SDA under-utilise the processing power on offer.
- The sudden deterioration in performance of the MVA for prediction horizons above 4, suggests that the size of the matrix  $\mathbf{V}$  becomes excessive and as a result impedes memory transfer and the processing speed of the CPU.

---

<sup>16</sup>INTEL<sup>©</sup> dual core i5-6200U, 2.30GHz processor with four threads per core.

The SDA has timorously been used in a multiple of practical applications [93, 95, 94] due to its simplicity and ease of implementation. However, the sequential nature of the branch and bound procedure combined with the low complexity per iteration, under-engage the resources on offer in modern processors. In the next chapter, this research will attempt to increase the complexity of the SDA to facilitate matrix processing, reduce memory traffic and introduce aspects of preprocessing. Not to discard the MVA completely, the next chapter will also consider the MVA's unique traits and attempt to find a fitting power electronic application.

## Chapter 5

# Alternative sphere decoder

Modern processors typically have more than one core allowing for fast matrix multiplication in the form of block matrix operations. These operations do not reduce the flop count, but they can dramatically affect the performance of the algorithm because of the way matrix data are handled in memory [115]. When a matrix becomes relatively large, the matrix size exceeds the cache memory size and must then partially be stored in slower memory. The result is a decline in processing speed and hence a drop-off in performance. This partially explain the behaviour of the MVA where superior performance was experienced in solving the CVPs of dimensions  $d \leq 12$ , and the sudden deterioration in higher dimensions.

The MVA can be described as a fast deterministic algorithm that unfortunately demands much resources from the processing hardware, especially in higher dimensions. In an attempt to develop an improved algorithm that best marries moderate complexity with hardware resources, this research proposes to adapt the conventional SDA to the specific CVP problem that originates from FCS-MPC of power electronic systems. The algorithm presented hereafter will equip the conventional SDA with matrix operations for the purpose of engaging the parallel processing capabilities of the hardware. These operations will inadvertently increase the complexity or flop count of the algorithm. To mitigate the unavoidable increase, the matrix operations will be confined to manageable blocks and also be supported by some offline preprocessing. Accordingly, the proposed algorithm will be referred to as the *Sphere Block Decoding algorithm* (SBDA).



## 5.1 Sphere block decoding

In section 3.5.1, the SDA was described as a process of pruning a search tree of depth  $d$  (dimensionality of the CVP) with branching at every dimensional level corresponding to the elements of the control set  $\mathcal{U}$ . This approach is successful in minimising the SDA's complexity per iteration, as the cost of a one-dimensional branching step is computed and weighed against the sphere radius. Unfortunately, this approach allows for a high upper bound in terms of algorithm iterations, and also inhibits the parallel processing capabilities of the hardware. To engage the processing hardware in a more effective way, some matrix calculations are suggested.

### 5.1.1 Concept

The question that follows is what size of matrix should be decided upon. Particular to the *three-phase*, multilevel inverters used in this work, the size of the generator matrix  $\mathbf{H} \in \mathbb{R}^{d \times d}$ , is a function of the prediction horizon  $N$ , and the dimensionality  $d_u$ , of the inverter control vector  $\mathbf{u}_{abc}$  (2.45). For the NPC inverter, specifically,

$$d = Nd_u = 3N. \quad (5.1)$$

An ideal hardware environment would enable a decoder to compute in time and with a single iteration, the cost of *all* feasible control sequences  $\tilde{\mathbf{u}} \in \mathbb{U}$  and extract the sequence  $\hat{\mathbf{u}}$  with minimal cost. For current processing technology this amount of complexity is an impossible task, especially for large values of  $d$  which sets the feasible set cardinality to  $\#\mathbb{U} = 3^d$ . Hence, the necessity for a decoding process such as the SDA but with some compromise. The most obvious and versatile option stems from (5.1) where the dimensionality of a single prediction horizon step is dictated by the dimensionality of the inverter control vector  $d_u = 3$ . It is therefore proposed to incorporate three-dimensional operations per algorithm iteration. Adopting the SDA approach and augmenting it to consider three-dimensional vectors per iteration requires reformulation of the search tree. Accordingly, this defines the search tree with the levels designated by the integer horizons 1 to  $N$ . At every tree level, the respective parent nodes branch into  $\#\mathcal{U} = 27$  three-dimensional sibling nodes. Figure 5.1 illustrates the principle by means of a random control sequence traversed through the reconfigured tree structure with depth  $N$  (left),

and the conventional tree structure with depth  $d$  (right). The example is typical to the FCS-MPC problem of a three-phase, three-level inverter with prediction horizon  $N = 3$ .

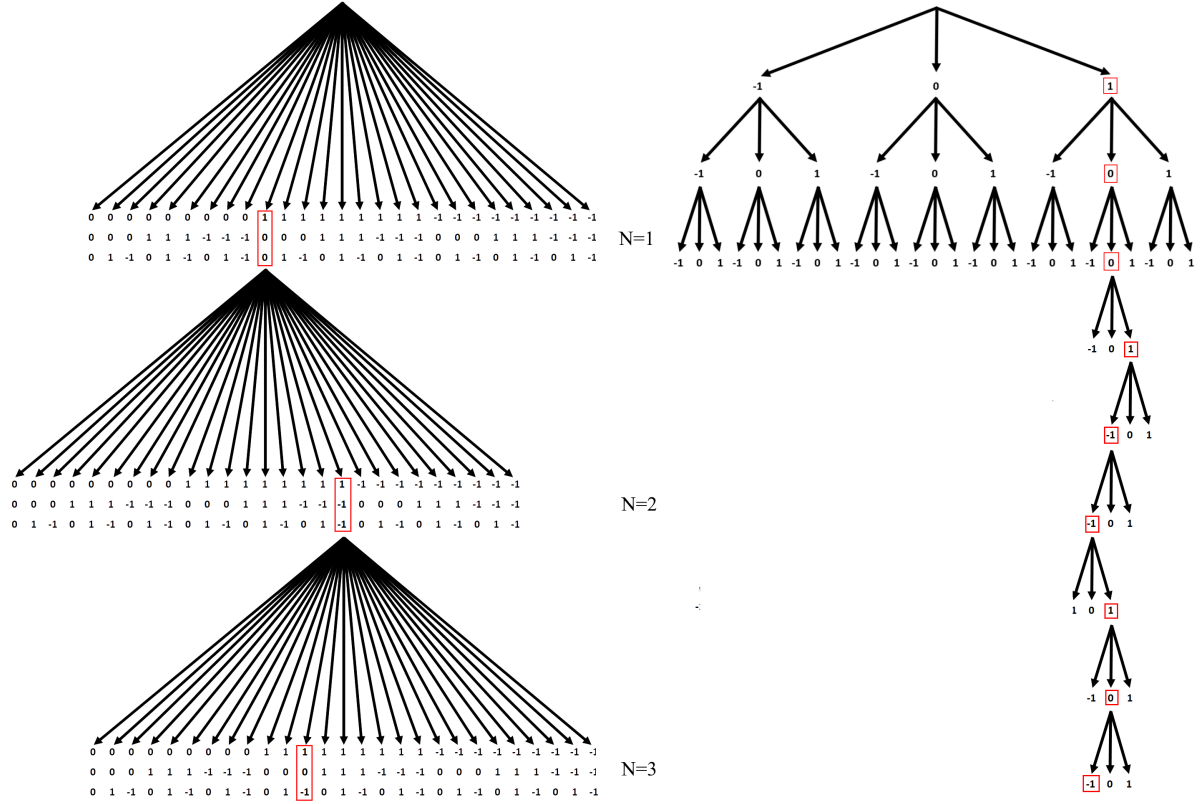


Figure 5.1: Tree traversal through a three-dimensional tree structure (left) and one-dimensional tree structure (right) to the 3<sup>rd</sup> horizon / 9<sup>th</sup> dimension.

The proposed strategy deviates from the conventional approach where at every parent node, a single one-dimensional branching step is evaluated to a process where 27 three-dimensional branches are evaluated simultaneously. The branches that effect a partial cost less than the sphere radius identifies the siblings which are to be considered as parents in the next tree level (horizon). In a similar fashion as the conventional SDA, the process follows a depth-first approach and prunes the branches with excessive cost. The search ends when only one path through it remains, i.e. the optimal solution

$$\hat{\mathbf{u}} = [\hat{\mathbf{u}}_1^T \hat{\mathbf{u}}_2^T \dots \hat{\mathbf{u}}_N^T]^T \in \mathbb{Z}^d, \quad (5.2)$$

which consists of  $N$ , three-dimensional branches or sub-vectors

$$\hat{\mathbf{u}}_n \in \mathcal{U}, \quad n = 1, 2, \dots, N. \quad (5.3)$$

As a first observation, this approach seems demanding in terms of computational effort. However, it allows for precomputation of the branching costs in the search tree. Compu-

tation of these costs necessitates the representation of the inverter control set  $\mathbf{U}$  as the matrix

$$\mathbf{U}_{abc} = [\mathbf{u}_{abc,1} \ \mathbf{u}_{abc,2} \dots \mathbf{u}_{abc,27}] \in \mathbb{Z}^{3 \times 27}. \quad (5.4)$$

Specific to this work, the 27 control vectors ( $\mathbf{u}_{abc,p}$  with  $p = 1, 2, \dots, 27$ ) are indexed as the column vectors of the matrix:

$$\mathbf{U}_{abc} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 1 & 1 & 1 & -1 & -1 & -1 & 0 & 0 & 0 & 1 & 1 & 1 & -1 & -1 & -1 & 0 & 0 & 1 & 1 & 1 & -1 & -1 \\ 0 & 1 & -1 & 0 & 1 & -1 & 0 & 1 & -1 & 0 & 1 & -1 & 0 & 1 & -1 & 0 & 1 & -1 & 0 & 1 & -1 & 0 & 1 & -1 & -1 \end{bmatrix}.$$

Transformation of these three-dimensional control vectors to the  $\mathbf{H}$ -coordinate space can be accomplished with three-by-three transformation matrices. Hence, the partitioning of matrix  $\mathbf{H} \in \mathbb{R}^{d \times d}$  into sub-matrices  $\mathbf{H}_z \in \mathbb{R}^{3 \times 3}$ . The result totals a number of  $N^2$  blocks, of which

$$y = \frac{N(N+1)}{2}, \quad (5.5)$$

are non zero sub-matrices,

$$\mathbf{H}_z, \ z = 1, 2..y, \quad \mathbf{H}_z \subset \mathbf{H}. \quad (5.6)$$

Numbering of the non-zero sub-matrices in this work, is done in a top-to-bottom, and left-to-right manner, as demonstrated for the horizon  $N = 3$  case,

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_1 & 0 & 0 \\ \mathbf{H}_2 & \mathbf{H}_3 & 0 \\ \mathbf{H}_4 & \mathbf{H}_5 & \mathbf{H}_6 \end{bmatrix}.$$

Note that each of the sub-matrices  $\mathbf{H}_z$  are square, three-by-three dimensional, and they transform the set of control vectors  $\mathbf{U}_{abc}$  to corresponding three-dimensional sub-vectors  $\mathbf{H}_z \mathbf{U}_{abc}$ . Also note that the rows of sub-matrices in  $\mathbf{H}$  are congruent to the search tree levels, i.e. intermediate horizons. This structure allows for the efficient precomputation and storage of the sub-vectors in sub-matrices

$$\mathbf{G}_z = \mathbf{H}_z \mathbf{U}_{abc}, \ z = 1, 2..y, \quad \mathbf{G}_z \subset \mathbf{G}, \quad (5.7)$$

with the matrix  $\mathbf{G} \in \mathbb{R}^{3 \times 27 \times y}$  and  $y$  denoting the number of non-zero sub matrices. The online availability of the matrix  $\mathbf{G}$  readily simplifies the computational effort, as can be

shown by the following example. Consider the control sequence depicted in Figure 5.1 as the optimal solution to the CVP, with target vector

$$\mathbf{x}^* = [\mathbf{x}_1^{*T} \ \mathbf{x}_2^{*T} \ \dots \ \mathbf{x}_N^{*T}]^T \in \mathbb{R}^d, \quad (5.8)$$

expressed in terms of three-dimensional sub-vectors

$$\mathbf{x}_n^* \in \mathbb{R}^3, \quad n = 1, 2, \dots, N. \quad (5.9)$$

The cost of traversing the 27 branches from the root node to the first tree level ( $n = 1$ ) is obtained from augmenting (2.35c) to give

$$\tilde{\mathbf{J}}_n = \|\mathbf{H}_1 \mathbf{U}_{abc} - \mathbf{x}_n^* \mathbf{o}^T\|_C^2 \quad (5.10a)$$

$$= \|\mathbf{G}_1 - \mathbf{x}_n^* \mathbf{o}^T\|_C^2. \quad (5.10b)$$

Note that vector  $\mathbf{o} = [1_1 1_2 \dots 1_{27}]^T$ , and that the outer product with the sub-vector  $\mathbf{x}_n^*$ , results in a matrix with recurring column vectors of  $\mathbf{x}_n^*$ . The element-wise subtraction from  $\mathbf{G}_1$  leaves a  $\mathbb{R}^{3 \times 27}$ -matrix in the denotation  $\|\cdot\|_C$ . This function is introduced as a vector of Euclidean norms

$$\|\mathbf{A}\|_C = \left( \sum_{i=1}^k |a_{ij}|^2 \right)^{\frac{1}{2}} = [\|\mathbf{a}_1\|_2 \ \|\mathbf{a}_2\|_2 \ \dots \ \|\mathbf{a}_l\|_2] \quad (5.11)$$

$$\text{where } j = 1, 2, \dots, l \quad \text{and} \quad \mathbf{A} \in \mathbb{R}^{k \times l}. \quad (5.12)$$

The result from (5.10b) is a row vector with the elements thereof representative of the individual branching costs

$$\tilde{\mathbf{J}}_n = [\tilde{J}_1 \ \tilde{J}_2 \ \dots \ \tilde{J}_{27}] \in \mathbb{R}^{27}. \quad (5.13)$$

Traversing the tree levels also requires the introduction of an index vector

$$\mathbf{p} = [p_1 \ p_2 \ \dots \ p_N]^T \in \mathbb{Z}^N, \quad (5.14)$$

where each element refers to the corresponding branching steps per tree level. As per the example (Fig. 5.1), the branching to tree level ( $n = 1$ ) is identified by  $p_n = 10$ , i.e.  $\mathbf{u}_{abc, p_n} = [1 \ 0 \ 0]^T$ , with cost  $J_{p_n}$ . From this node, the investigation continues and considers the branchings to tree level ( $n = 2$ ). The subsequent cost computation resembles

$$\tilde{\mathbf{J}}_n = \|(\mathbf{H}_3 \mathbf{U}_{abc} + \mathbf{H}_2 \mathbf{u}_{abc, p_{n-1}} \mathbf{o}^T) - \mathbf{x}_n^* \mathbf{o}^T\|_C^2 + J_{p_{n-1}} \quad (5.15a)$$

$$= \|\mathbf{H}_3 \mathbf{U}_{abc} - (\mathbf{x}_n^* - \mathbf{H}_2 \mathbf{u}_{abc, p_{n-1}}) \mathbf{o}^T\|_C^2 + J_{p_{n-1}} \quad (5.15b)$$

$$= \|\mathbf{G}_3 - (\mathbf{x}_n^* - \mathbf{G}_{2, p_{n-1}}) \mathbf{o}^T\|_C^2 + J_{p_{n-1}}. \quad (5.15c)$$

Note that  $\mathbf{G}_{2,p_{n-1}}$  denotes the  $p_1^{th}$  column vector in matrix  $\mathbf{G}_2$ , and  $J_{p_{n-1}}$  the cost of traversal to level ( $n = 1$ ) via the  $p_1^{th}$  branch. In the example (Fig. 5.1), the branch to level ( $n = 2$ ) is identified by  $p_n = 18$  i.e.  $\mathbf{u}_{abc,p_n} = [1 \ -1 \ -1]^T$ , with cost  $J_{p_n}$ . Traversal from this node to the final tree level ( $n = 3$ ) result in the costs

$$\tilde{\mathbf{J}}_n = \|\mathbf{G}_6 - (\mathbf{x}_n^* - (\mathbf{G}_{5,p_{n-1}} + \mathbf{G}_{4,p_{n-2}}))\mathbf{o}^T\|_C^2 + J_{n-1,p_{n-1}}. \quad (5.16)$$

With reference to (5.10b), (5.15c) and (5.16), the evolution of the cost computation can be envisioned. The fundamental ( $3 \times 27$ )-matrix subtraction remains throughout for  $n \geq 1$ . For tree levels  $n > 1$ , the previous partial cost is included in the form of a ( $3 \times 27$ )-matrix addition. Updating of the sub-vector  $\mathbf{x}_n^*$  with the effect of previous branchings incurs ( $n - 1$ ) three-dimensional subtractions. For the purpose of stating the cost computation in terms of the tree level  $n$ , we introduce the variable

$$b = \frac{n(n+1)}{2}, \quad (5.17)$$

which allows for definition of the generic equation

$$\tilde{\mathbf{J}}_n = \|\mathbf{G}_b - (\mathbf{x}_n^* - (\mathbf{G}_{b-1,p_{n-1}} + \mathbf{G}_{b-2,p_{n-2}} + \dots + \mathbf{G}_{b-(n-1),p_1}))\mathbf{o}^T\|_C^2 + J_{n-1,p_{n-1}}. \quad (5.18)$$

The availability of matrix  $\mathbf{G}$  eliminates the need for online transformation computations, and as a result, minimises the computational effort to the element-wise subtraction and addition of a ( $3 \times 27$ )-matrix. This constitutes the bulk of the computational burden, with addition of the variable ( $n - 1$ ) three-dimensional vector subtractions.

## 5.2 Algorithm

The concept of the proposed sphere block decoding approach supported with aspects of preprocessing is presented in the following section. Similar to the conventional SDA, the SBDA outputs the optimal control sequence  $\hat{\mathbf{u}}$  upon input of a target  $\mathbf{x}^*$  and the estimated initial sphere radius  $\delta$ . In addition, the SBDA also requires the switching control set (5.4) and the precalculated matrix (5.7). Algorithm 6 lists the main routine SPHBLKDEC and Algorithm 7 the subroutine DECODE. The main routine initiates the subroutine (line:7), and upon receiving an index vector (5.14), constructs the optimal control sequence with elements of the index vector identifying the control vectors from the set  $\mathbf{U}_{abc}$ .

The subroutine DECODE initiates with the first tree level  $n = 1$ , the index vector  $\mathbf{p} \leftarrow \text{ones}(1, N)$  and a queue-vector  $\mathbf{q}$ , loaded with a single element referencing the root

---

**Algorithm 6** Sphere Block Decoding algorithm - main routine

---

```

1: function SPHBLKDEC( $\mathbf{x}^*$ ,  $\delta^2$ ,  $\mathbf{G}$ ,  $\mathbf{U}_{abc}$ )
2:    $n \leftarrow 1$ 
3:    $\mathbf{p} \leftarrow \text{ones}(1, N)$ 
4:    $\mathbf{q} \leftarrow [1]$ 
5:    $\tilde{\mathbf{J}} \leftarrow \text{zeros}(1, 27)$ 
6:    $\mathbf{o} \leftarrow \text{ones}(1, 27)$ 
7:    $[\hat{\mathbf{p}}, \delta^2] = \text{DECODE}(\mathbf{x}, \delta^2, \mathbf{G}, n, \mathbf{p}, \mathbf{q}, \tilde{\mathbf{J}}, \mathbf{o})$ 
8:   for  $n = 1 : N$  do
9:      $\hat{\mathbf{u}}_n \leftarrow \mathbf{u}_{abc, \hat{p}_n}$ 
10:  end for
11:  return  $\hat{\mathbf{u}}$ 
12: end function

```

---

node. Starting at the root necessitates the zeroing of the branching costs  $\tilde{\mathbf{J}}$ . As a first step the variable  $b$  is computed for the current tree level, followed by the for-loop, which repeats for the *number*<sup>19</sup> of admissible parent nodes listed in the queue-vector. The  $n^{\text{th}}$  level sub-vector is extracted from the target vector (line:4), followed by the gathering of the preceding branching cost. This is zero for tree level ( $n = 1$ ), and updating of the target sub-vector  $\mathbf{x}$  is also not required (lines:8 to 13). All 27 branching costs are computed in (line:14) and compared with the sphere radius (line:15). The resulting queue-vector reflects the index values of admissible branchings which are to be considered as parent nodes for branching to the next tree level. If admissible branchings exist, traversal to the next tree level is considered by re-calling the DECODE subroutine (line:21).

The routine is initiated with the new tree level ( $n + 1$ ), the branching path  $\mathbf{p}$ , the admissible parent nodes  $\tilde{\mathbf{q}}$  and the cost of all 27 traversals  $\tilde{\mathbf{J}}$ . Note that  $\mathbf{p}$ ,  $\tilde{\mathbf{q}}$  and  $\tilde{\mathbf{J}}$  convey the relative conditions at tree level  $n$ . The same sequence of events are followed whereby  $b$  is computed, the for-loop identifies the first parent node in the queue, and the target sub-vector is extracted (lines:1 to 4). Tree levels ( $n > 1$ ), require updating of the branching path with the selected parent node and also the branching cost incurred by it (lines:8 and 9). Updating of the target sub-vector with the effects of previous branchings (lines:10 to 12) again leads to the cost computation and identification of admissible branchings. If no viable branchings exists  $\tilde{\mathbf{q}} = []$ , the branch is pruned and the next parent in the queue  $\mathbf{q}$  is considered. Branching continues until the final tree level ( $n = N$ ) is reached, whereupon the branching with minimal cost is identified. The last element of

---

<sup>19</sup>The MATLAB function “nummel” returns the number of elements present in a vector.

**Algorithm 7** Sphere Block Decoding algorithm - subroutine

---

```

1: function DECODE( $\mathbf{x}^*, \delta^2, \mathbf{G}, n, \mathbf{p}, \mathbf{q}, \tilde{\mathbf{J}}, \mathbf{o}$ )
2:    $b \leftarrow \text{sum}(1 : n)$ 
3:   for  $m = 1 : \text{nummel}(\mathbf{q})$  do
4:      $\mathbf{x} \leftarrow \mathbf{x}_n^*$ 
5:     if  $n = 1$  then
6:        $J \leftarrow \tilde{J}_{p_n}$ 
7:     else
8:        $p_{(n-1)} \leftarrow q_m$ 
9:        $J \leftarrow \tilde{J}_{p_{n-1}}$ 
10:      for  $y = 1 : (n - 1)$  do
11:         $\mathbf{x} = \mathbf{x} - \mathbf{G}_{(b-y), p_{(n-y)}}$ 
12:      end for
13:    end if
14:     $\tilde{\mathbf{J}} = \|\mathbf{G}_b - \mathbf{x}\mathbf{o}\|_C^2 + J$ 
15:     $\tilde{\mathbf{q}} = \text{find}(\tilde{\mathbf{J}} \leq \delta^2)$ 
16:    if  $\text{nummel}(\tilde{\mathbf{q}}) \geq 1$  then
17:      if  $n = N$  then
18:         $[\delta^2, p_n] = \text{min}(\tilde{\mathbf{J}})$ 
19:         $\hat{\mathbf{p}} \leftarrow \mathbf{p}$ 
20:      else
21:         $[\hat{\mathbf{p}}, \delta^2] = \text{DECODE}(\mathbf{x}^*, \delta^2, \mathbf{G}, (n + 1), \mathbf{p}, \tilde{\mathbf{q}}, \tilde{\mathbf{J}}, \mathbf{o})$ 
22:      end if
23:    end if
24:  end for
25:  return  $[\hat{\mathbf{p}}, \delta^2]$ 
26: end function

```

---

the index vector  $p_n$  is updated, and the branching cost incurred for the recorded path  $\mathbf{p}$  is taken as the sphere radius. With the tightened sphere radius the algorithm continues to prune non-admissible branches while searching for more cost-effective paths, until a single optimal path  $\hat{\mathbf{p}}$  remains. This path is returned to the *main routine*, whereupon the optimal control sequence  $\hat{\mathbf{u}}$  is reconstructed.

Although not listed in Algorithm 7, the queue of admissible parent nodes at each tree level can be pruned further by applying the *switching constraint* and/or other selection strategies like *best-first* sorting.

### 5.2.1 Complexity

The proposed tree structure of the SBDA is clearly more complex than the conventional approach, and an increase in the computational burden per algorithm *iteration* is to be

expected. Computation of the 27 branching costs (line:14) contributes to the bulk of the burden imposed by Algorithm 7. It amounts to a  $(3 \times 27)$ -matrix which undergoes subtraction, element squaring and column summations. Element wise, the total number of operations for (line:14) can be conveyed as

$$(3 \times 27)(2 \otimes + \oplus + \ominus). \quad (5.19)$$

To a lesser extend, updating of the target sub-vector (line 11) requires

$$(3n - 3)\ominus \quad (5.20)$$

element wise operations. Cumulatively, (5.19) and (5.20) amount to

$$3n + 321 \quad \text{flops} \quad (5.21)$$

or real-time operations per parent node visited. The number of node visits, i.e. algorithm iterations for the sphere decoding approach, are dependent on the selected sphere-radius. An optimal sphere radius selection that includes only one lattice point (the optimal solution), will result in a minimum number of algorithm iterations, which equals the depth of the relevant search tree. For the SBDA with tree depth  $n = N$ , the lower computational bound is therefore set to

$$\frac{(3N^2 + 645N)}{2} \quad \text{flops.} \quad (5.22)$$

In comparison to the conventional SDA (3.45), Figure 5.2 illustrates the theoretical lower bounds in *flop count* of the respective algorithms. Note that the SBDA poses a higher flop count at low horizons, but gains on the SDA as the prediction horizon increases. Importantly, the results shown in Figure 5.2 are a *direct* comparison of the computational efforts exerted by the algorithms for the *best-case scenario* in terms of sphere-radius selection. The results highlight the *fundamental nature* of the SBDA as a direct result of the offline preprocessing, i.e. the sub-vectors in matrix  $\mathbf{G}$ . Any gains that *may* arise from fast matrix operations are not conveyed in these results.

Storage of the matrix  $\mathbf{G}$  demands memory capacity, which can be determined with

$$C = d_b \times \left( \sum_{n=1}^{d/d_b} n \right) \times d_u^{d_b} \times 8 \text{ bytes}, \quad (5.23)$$



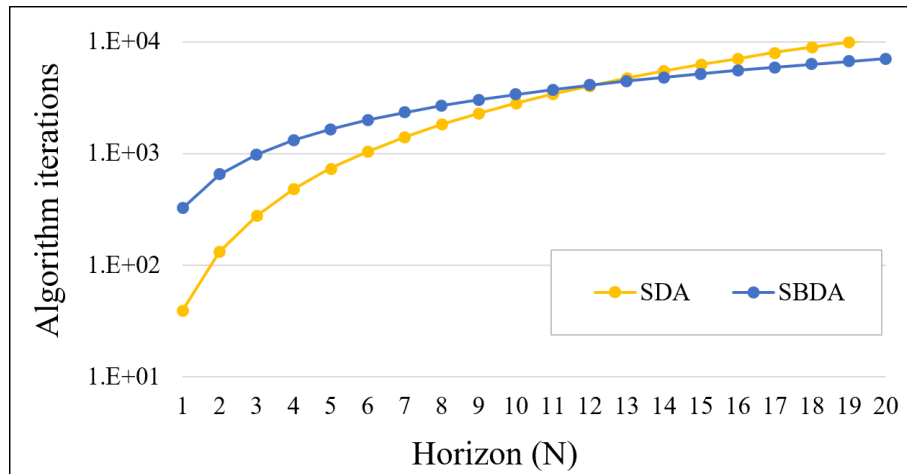


Figure 5.2: Computational lower bounds for the SDA and SBDA when solving MPC problems of horizon  $N$ .

where  $d_b$  denotes the dimension of the blocks or sub-matrices (5.6) which  $\mathbf{H}$  is partitioned into. The second term defines the number of sub-matrices, and the third reflects the cardinality of a congruent  $d_b$ -dimensional control vector set. For storage implications, a word length of 8 bytes is assumed. Consider an example with a prediction horizon of  $N = 12$ , which effects a generator matrix  $\mathbf{H} \in \mathbb{R}^d$ ,  $d = 3N$ . As per the SBDA concept presented above, the matrix  $\mathbf{H}^{36 \times 36}$  can be partitioned into three-dimensional sub-matrices, and as a result be represented by 78 non-zero sub-matrices. Each of these matrices generates 27 three-dimensional sub-vectors which are to be stored in matrix  $\mathbf{G}$ . This effects a storage demand of  $3 \times 27 \times 78 \times 8 \text{ bytes} = 51 \text{ kB}$ .

Naturally, the question arises: why not use a different block structure? The answer is three-fold. Firstly, the block size ( $d_b$ ) must facilitate the MPC prediction horizon selection  $3N/d_b \in \mathbb{Z}$ ; secondly, it should not be too much of a computational challenge in terms of matrix size ( $d_b \times d_u^{d_b}$ ), and thirdly, it results in reasonable storage capacity. In comparison, consider the listing for different block sizes in Table 5.1. A common prediction

Table 5.1: Influence of block-size selection for a 3-level inverter.

$d$	$d_b$	$d_u^{d_b}$	$y$	kB	$N$
36	1	3	666	16	1, 2, ...
36	2	9	171	25	2, 4, ...
36	<b>3</b>	<b>27</b>	<b>78</b>	<b>51</b>	1, 2, ...
36	4	81	45	117	4, 8, ...
30	5	243	21	204	5, 10, ...
36	6	729	21	735	2, 4, ...

horizon of  $N = 12$  is considered for all the different block sizes, except for the  $d_b = 5$  case, where  $36/5 \notin \mathbb{Z}$ . This limitation is also expressed in the last column, which denotes the prediction horizons that can be facilitated. Note that only block sizes of one and three facilitate all prediction horizon options.

### 5.3 Performance simulations

Fundamentally, the proposed SBDA remains a decoding algorithm based on the sphere principle of [41]. Accordingly, the same aspects apply in terms of sphere radius selection and the effect it has on the algorithm iterations. As the search space of the CVP will vary for different systems, the performance of a specific decoding algorithm can be estimated through simulation of the system under various conditions. These conditions will influence the location of the CVP target, and as a consequence the initial sphere radius, and ultimately the performance of the decoding algorithm.

MATLAB<sup>®</sup> simulations of the NPC inverter driving an RL-load were repeated with the steady-state and transient conditions as defined in section 4.2.1 to 4.2.2. The simulation process illustrated by the state-machine in Fig. 4.2 was followed with the SBDA inserted in the decoder algorithm block. In the results following, note that the performance of the SBDA combined with the projection algorithm will be referred to as the SBDap strategy. The simulation results recorded for the SBDap during steady-state and transient conditions are displayed in Figure 5.3 and Figure 5.4 respectively.

In steady-state conditions, and as expected, the SBDap exhibits a higher computational complexity than the SDap in lower horizons, but it improved as the prediction horizon extended. The maximum algorithm termination times for the SBDap were the fastest for horizons seven and higher, even outperforming the conventional SDA with least complexity. In transient conditions the SBDap and SDap reaped the rewards from target preconditioning, as their maximum flop counts remained approximately the same. Even though the SBDap recorded slower termination times than the MVAp for prediction horizons  $N \leq 4$ , the simulation results indicate it to be the better option for longer prediction horizons.

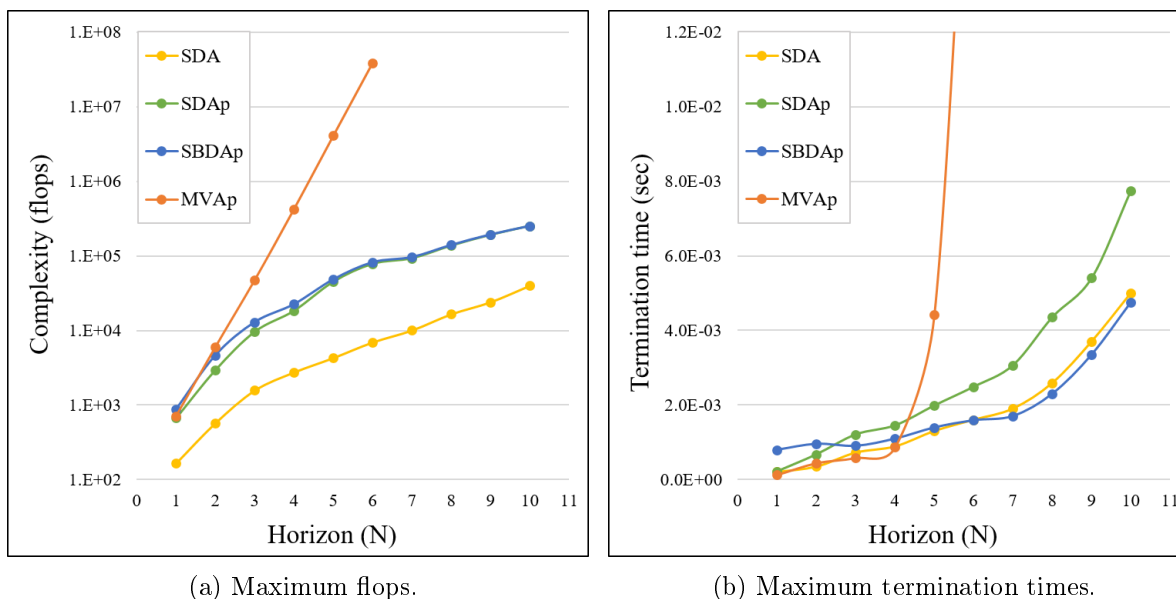


Figure 5.3: Performance of the decoding algorithms in steady-state conditions. The sphere decoding algorithm without target preconditioning is denoted by SDA. Performance of the SDA, MVA and SBDA *combined with* the projection algorithm, are respectively denoted by SDAP, MVAP and SBDAP.

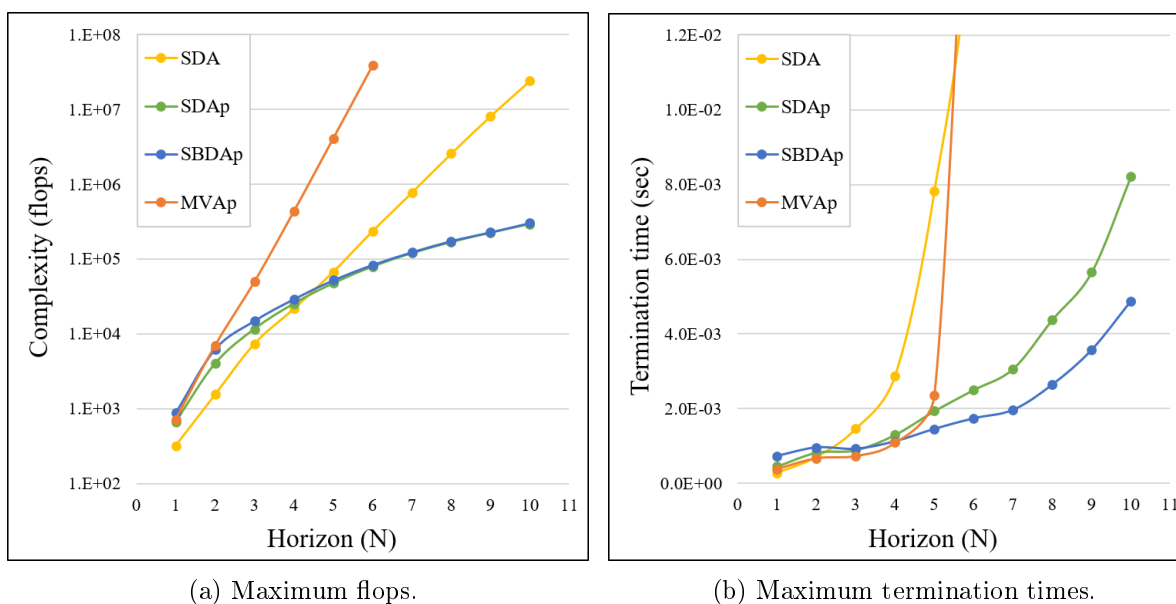


Figure 5.4: Performance during transient conditions. The sphere decoding algorithm without target preconditioning is denoted by SDA. Performance of the SDA, MVA and SBDA *combined with* the projection algorithm, are respectively denoted by SDAP, MVAP and SBDAP.

## 5.4 Multilevel considerations

Control of an multilevel inverter by means of FCS-MPC typically results in a CVP where the number of lattice points *per dimension* is governed by the cardinality of the inverter-leg control set  $\mathcal{U}$  (2.44). This is a direct result of the number of DC voltage levels  $l_v$  employed by the multilevel inverter topology. The simulation results presented above, estimated the performances of the respective decoding strategies for extended prediction horizons. To further the investigation, the decoding algorithms are hereafter considered for solving the CVPs related to the FCS-MPC of multilevel inverter topologies with varying voltage levels. Use of the cascaded H-bridge topology requires the system model in Section 4.1.1 to be augmented in terms of the inverter output voltage (4.2). From (2.44), the inverter output voltage is re-defined as

$$\mathbf{v}_i = [v_{i\alpha} \ v_{i\beta}]^T = v_{dc} \mathbf{K} \mathbf{u}_{abc}, \quad \mathbf{u}_{abc} \in \mathcal{U}. \quad (5.24)$$

For the three-phase case ( $d_u = 3$ ), an increase in the inverter voltage levels ( $l_v$ ) has no effect on the dimensionality of the relevant CVP problem  $d = Nd_u$ . However, the cardinality of the inverter control set and the set of feasible control sequences for optimisation are respectively given by  $\#\mathcal{U} = l_v^3$  and  $\#\mathbb{U} = l_v^{3N}$ . To visually demonstrate this increase in lattice structure, Figure 5.5 shows the first *two dimensions* of the search space typical to the FCS-MPC problem of a five-level inverter with prediction horizon  $N = 3$ . Note

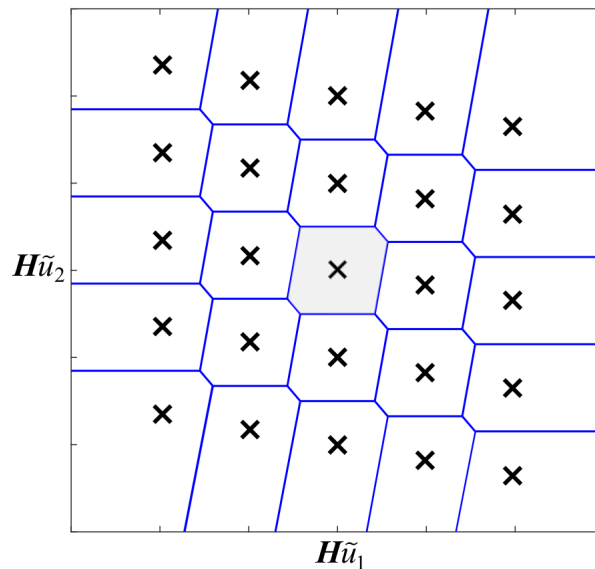


Figure 5.5: Lattice structure in two dimensions of the solution space for the FCS-MPC problem of a five-level diode-clamped multilevel inverter.

the basic Voronoi cell of the origin and its translation to the other lattice points.

MATLAB<sup>®</sup> simulations were again conducted as defined in Section 4.2.1 to 4.2.2. Prediction horizons of  $N = 3, 4$  and multilevel inverter topologies with voltage levels  $l_v = 3, 5 \dots 11$  were considered. The conventional SDA without target preconditioning was disregarded due to its disadvantage in transient conditions. Note that the projection notation “p” in SDAp, MVAp and SBDAp will be discarded henceforth, and that reference to any CVP solver implies the *working from a preconditioned target*  $\xi^*$ . Projection onto the lattice convex hull remained in principle as before, except that the outer-bound of the lattice needs to be adjusted with the factor  $b = (l_v - 1)/2$ . The inclusion of  $l_v$  together with the necessary adjustments to the projection algorithm’s pseudo code is listed in Appendix B.1. Figure 5.6 and Figure 5.7 plot the MATLAB<sup>®</sup> simulation results for the different solvers during steady-state operation of the power electronic system with multilevel inverters varying in voltage level numbers. The performance metrics reported on are algorithm complexity (maximum flops) and maximum termination time ( $t_c$ ).

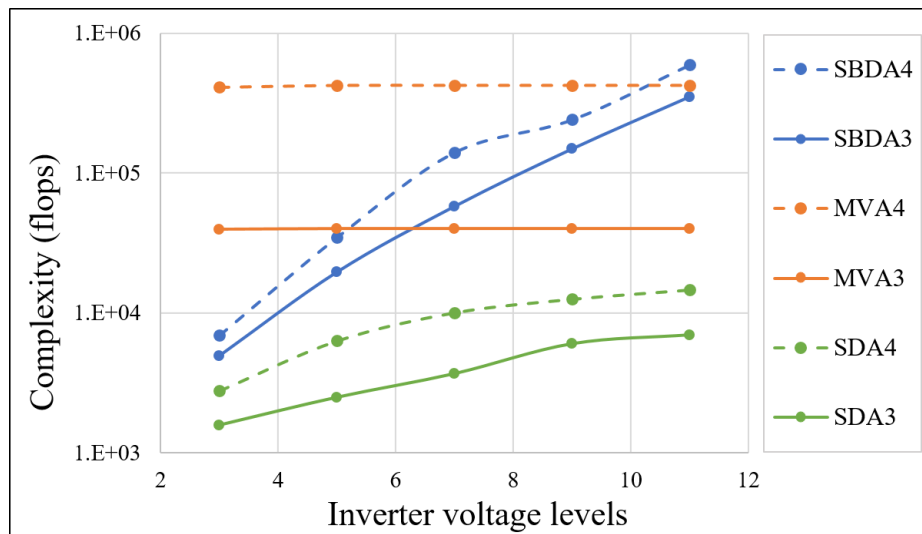


Figure 5.6: Maximum flop counts effected by the CVP solvers for a number of multilevel inverters with voltage levels ranging from three to eleven. MPC prediction horizons of  $N = 3$  and 4 are shown.

As expected from previous simulation results, the SDA exhibits the lowest complexity for both prediction horizons. The complexity of all three solvers are negatively affected by an increase in the prediction horizon. Noteworthy is the observation that, the increase in inverter voltage level leaves the MVA’s complexity unaffected. The general negative effect of increasing the MPC prediction horizon and inverter voltage levels also manifests

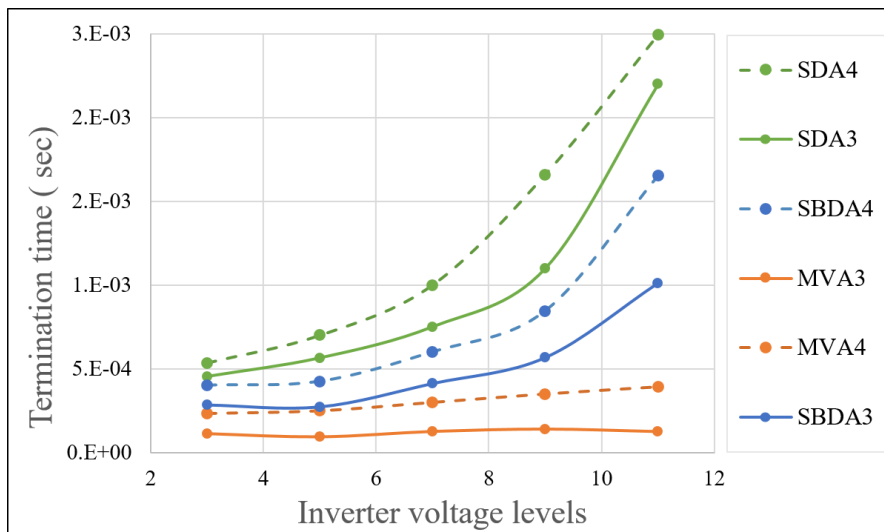


Figure 5.7: Maximum termination times of the algorithms to solve the optimization problem underlying MPC for prediction horizons of  $N = 3$  and 4.

in the algorithm termination times shown in Figure 5.7. Interestingly, the MVA recorded the fastest termination times for both prediction horizons over the entire range of inverter voltage levels. Due to the use of a preconditioned target that minimises the effect of transients, the simulation results for these conditions are not repeated.

## 5.5 Summary

The issue of solving long horizon FCS-MPC problems particular to a power electronic system application was investigated in this chapter. An alternative to the conventional sphere decoding algorithm was proposed and analysed. Adapted for modern processors, the sphere block decoding algorithm attempts to increase the computational efficiency of the decoding process through a combination of offline preprocessing and online block-processing. The computational complexity of the SBDA was shown to become competitive as the MPC prediction horizon extended. In terms of termination times, the MATLAB timing function indicated that the SBDA is favoured for prediction horizons  $N \geq 5$ .

The Micciancio and Voulgaris algorithm remained the algorithm with the highest computational complexity and fastest termination times for prediction horizons  $N < 5$ . In the previous chapter (section 4.3), it was mentioned that the MVA is well equipped to solve the CVP in a recursive lattice structure. This is mainly due to its reliance on the basic Voronoi cell of a lattice which is not necessarily affected when the size of the lattice

increases. Upon consideration of these characteristics, the study furthered investigation to find a power electronic system that effects a CVP which can best marry the MVA. FCS-MPC problems with an enlarged search space, i.e. more *lattice points per dimension* led to the consideration of diode-clamped multilevel inverters with voltage levels exceeding three. Simulations confirmed the deterministic nature of the MVA as the computational complexity remained constant, even though the lattice search space increased.

All the results obtained to this point in the writing were from MATLAB<sup>®</sup> simulations of a relatively simple power electronic system. The simulations assisted in the development of the projection algorithm, the SBDA and exploration of the MVA's inherent characteristics. Although the respective complexities of the algorithms were determined, the termination times reported by the MATLAB<sup>®</sup> timing function are only an *indication* and not a real-time measure of algorithm performance. For this reason, the performance of the different algorithms are explored in real time in the next chapter.

# Chapter 6

## Real-time verification

### 6.1 Introduction

To support the implementation of long-horizon MPC of a power electronic system in real time, this research, in the previous chapters, proposed a projection algorithm and decoding algorithm to solve the underlying optimisation problem. Although the simulations incorporated an *RL*-load, i.e. time-invariant plant of first order, which is relatively simple to manage, the performance indicators suggested that improved efficiency and optimality are expected from the developed algorithms. In this chapter, the respective algorithms are incorporated into the FCS-MPC of a more complex, time-varying system in real time. The system to be considered is similar to the one presented in [24], where an induction machine is driven by a three-phase, three-level NPC inverter via an intermediate *LC*-filter. For such a system of higher order, it was concluded by [5] that long-horizon MPC is particularly beneficial, especially when switching is expensive, i.e. the required converter switching frequency is low. This type of power electronic system substantiates the use of long-horizon MPC and the subsequent need for an efficient solver to the accompanying optimisation problem.

The rest of this chapter will present the control system and plant model, followed by the MPC formulation and practical considerations. Hardware-in-the-loop results and subsequent discussion will conclude the chapter.



## 6.2 Case study: Induction machine drive

Existing fixed speed induction motors are commonly retrofitted with medium voltage variable speed drives so as to achieve energy savings [116]. In cases where the motors are not designed to be directly connected to a power electronic converter, it is necessary to incorporate an intermediate  $LC$  filter between drive and motor. The filter improves the sinusoidal shape of the stator voltage, which leads to less harmonic distortions in the stator windings, reducing isolation problems and harmful bearing currents [117, 118]. In this configuration, a reduction in converter switching frequency can become problematic as it approaches the resonance frequency of the  $LC$  filter [116]. To address this issue and prevent excitation of the filter's resonance, the use of a *prediction horizon period*

$$T_N = NT_s, \quad (6.1)$$

that covers a significant part of the filter's resonance period is required. Geyer (2016) [5] advises prediction horizons of  $N \geq 3$ .

## 6.3 Control system

The electrical drive system can be defined as a multiple-input multiple-output (MIMO) system managed by an FCS-MPC designed, MIMO controller. Figure 6.1 depicts the layout. Note that the system under control, i.e. the *plant*, is indicated with the dotted lines.

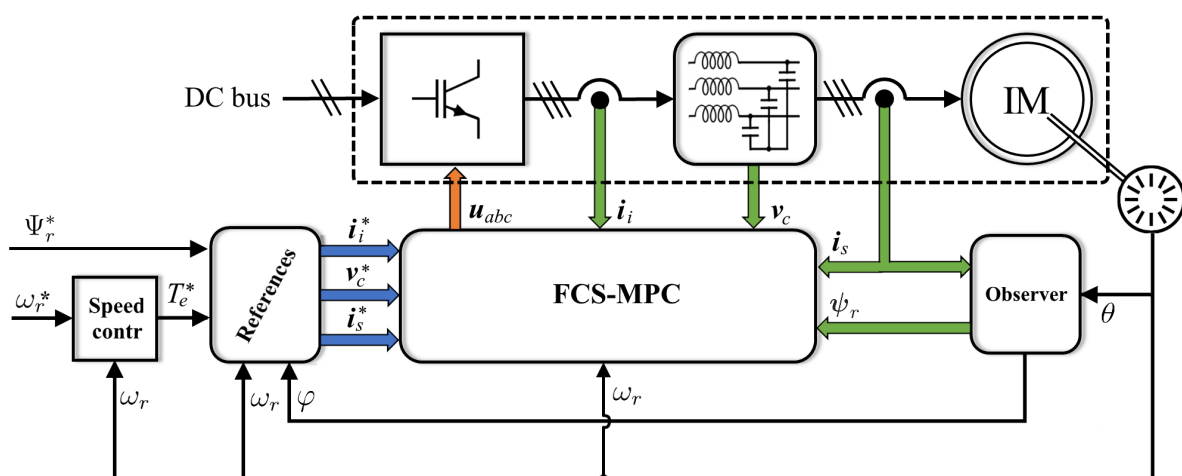


Figure 6.1: Three-phase three-level diode-clamped multilevel inverter driving an induction machine via an intermediate  $LC$  filter.

The general cascaded approach of a predictive current control system is followed where

the electromagnetic subsystem of the IM is managed by a predictive controller (FCS-MPC) and the mechanical subsystem by a proportional-integral (PI) speed controller. Noteworthy, the following assumptions are made:

- A three-phase, NPC inverter is considered with its capacitor voltages regulated and constant.
- Due to the relatively slow dynamics of the mechanical subsystem compared to the sampling frequency, the rotor speed  $\omega_r$  is assumed constant for the duration of the prediction horizon. This allows for considering  $\omega_r$  as a time-varying parameter.
- The plant sub-circuits and their respective equations are adopted from Section 2.5 with the per-unit parameters as listed in Table 6.1.

Table 6.1: Parameters of the drive system.

Parameter	Symbol	SI value	pu value
Nominal stator voltage	$V_R$	3300 V	0.5774
Nominal stator current	$V_R$	356 A	1.0000
Nominal power	$P_R$	1.587 MW	0.7799
Apparent power	$S_R$	2.035 MVA	1.0000
Rated stator frequency	$f_{sR}$	50 Hz	1.0000
Number of pole pairs	$p$	5	
Rotational speed	$\Omega_{mR}$	595 rpm	0.9913
Air-gap torque	$T_R$	26.2 kNm	1.0000
Stator resistance	$R_s$	57.61 m $\Omega$	0.0108
Rotor resistance	$R_r$	48.89 m $\Omega$	0.0091
Stator leakage inductance	$L_s$	2.544 mH	0.1493
Rotor leakage inductance	$L_r$	1.881 mH	0.1104
Main inductance	$L_m$	40.01 mH	2.3486
Filter inductance	$L$	2 mH	0.1174
Filter Capacitance	$C$	200 $\mu$ F	0.3363
Filter inductor resistance	$R_l$	2.0 m $\Omega$	0.0004
Filter capacitor resistance	$R_c$	2.0 m $\Omega$	0.0004
DC link voltage	$V_{dc}$	5200 V	1.9299

The objective of the controller is to manipulate the state of the converter legs  $\mathbf{u}_{abc}$ , such that the system outputs, i.e. inverter current  $\mathbf{i}_i$ , capacitor voltage  $\mathbf{v}_c$  and stator current  $\mathbf{i}_s$ , closely track their respective references (\*), while limiting the average converter switching frequency (2.51). System states are obtained through direct measurement and an observer. The system output references which define the operating point of the induction machine

are computed from the rotor flux  $\Psi_r^*$  setting and the torque requirement  $T_e^*$ . For optimal machine magnetisation,  $\Psi_r^*$  is typically held constant at its maximum value but can be decreased if field-weakening is required. Operating the induction machine at its *rated* values, requires a nominal rotor flux setting which correspond to a per unit stator flux magnitude of unity ( $\psi_s = 1\text{pu}$ ). Regulation of the rotor speed is managed by a typical PI speed controller which commands the torque requirement  $T_e^*$  from the speed error  $\omega_e = \omega_r^* - \omega_r$ . Rewriting (2.59) defines the speed equation

$$\frac{d\omega_r}{dt} = \frac{p(T_e - T_l)}{M}, \quad (6.2)$$

where the variables  $p, T_e, T_l$  and  $M$  respectively denote the IM number of pole pairs, the electromagnetic torque, the load torque and the combined inertia of the rotor and load. The electrical rotor speed  $\omega_r$  is computed from the angular position of the rotor shaft  $\theta$ , supplied by an incremental shaft encoder. The same information, together with the measured stator current allows for an observer to estimate the rotor flux  $\psi_r$  and its relative angular position  $\varphi$ . This estimation will be discussed in more detail at a later stage (see subsection 6.5.2.1).

## 6.4 Plant dynamics

In accordance with the pu-system, the plant dynamics in the stationary  $\alpha\beta$  coordinate system can be interpreted as follows. The three-dimensional control vector  $\mathbf{u}_{abc} \in \mathbf{U} \in \mathbb{Z}^3$  commands the three legs of the inverter which effects an output voltage

$$\mathbf{v}_i = [v_{i\alpha} \ v_{i\beta}]^T = \frac{V_{dc}}{2} \mathbf{K} \mathbf{u}_{abc}. \quad (6.3)$$

This acts as an input to the balanced three-phase *LC*-filter

$$\frac{d\mathbf{i}_i}{dt} = \frac{1}{L} (\mathbf{v}_i - R_l \mathbf{i}_i - \mathbf{v}_s) \quad (6.4a)$$

$$\frac{d\mathbf{v}_c}{dt} = \frac{1}{C} (\mathbf{i}_i - \mathbf{i}_s). \quad (6.4b)$$

A filter input current  $\mathbf{i}_i = [i_{i\alpha} \ i_{i\beta}]^T$  results, together with the filter output, i.e stator voltage

$$\mathbf{v}_s = [v_{s\alpha} \ v_{s\beta}]^T = \mathbf{v}_c + R_c (\mathbf{i}_i - \mathbf{i}_s). \quad (6.5)$$

This voltage acts upon the induction machine model

$$\frac{d\mathbf{i}_s}{dt} = -\frac{1}{\tau_s}\mathbf{i}_s + \left( \frac{1}{\tau_r} - \omega_r \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \right) \frac{L_m}{D}\boldsymbol{\psi}_r + \frac{L_r}{D}\mathbf{v}_s \quad (6.6a)$$

$$\frac{d\boldsymbol{\psi}_r}{dt} = \frac{L_m}{\tau_r}\mathbf{i}_s - \left( \frac{1}{\tau_r} - \omega_r \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \right) \boldsymbol{\psi}_r. \quad (6.6b)$$

to set up the stator current  $\mathbf{i}_s = [i_{s\alpha} \ i_{s\beta}]^T$ . Note that  $D$  refers to (2.62c), and that  $\omega_{fr}$  and  $\mathbf{v}_r$  are omitted from (2.65) due to the  $\alpha\beta$  notation and squirrel-cage construction of the induction machine. Substitution of  $\mathbf{v}_i$  and  $\mathbf{v}_s$  with (6.3) and (6.5) in (6.4) and (6.6) allows for defining the continuous-time differential equations of the plant

$$\frac{d\mathbf{i}_i}{dt} = \frac{-R_c - R_l}{L}\mathbf{i}_i - \frac{1}{L}\mathbf{v}_c + \frac{R_c}{L}\mathbf{i}_s + \frac{V_{dc}}{2L}\mathbf{K}\mathbf{u}_{abc} \quad (6.7a)$$

$$\frac{d\mathbf{v}_c}{dt} = \frac{1}{C}(\mathbf{i}_i - \mathbf{i}_s) \quad (6.7b)$$

$$\frac{d\mathbf{i}_s}{dt} = \frac{R_c L_r}{D}\mathbf{i}_i + \frac{L_r}{D}\mathbf{v}_c - \left( \frac{1}{\tau_s} + \frac{R_c L_r}{D} \right) \mathbf{i}_s + \left( \frac{1}{\tau_r} - \omega_r \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \right) \frac{L_m}{D}\boldsymbol{\psi}_r \quad (6.7c)$$

$$\frac{d\boldsymbol{\psi}_r}{dt} = \frac{L_m}{\tau_r}\mathbf{i}_s - \left( \frac{1}{\tau_r} - \omega_r \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \right) \boldsymbol{\psi}_r. \quad (6.7d)$$

Congruently and still in the  $\alpha\beta$  frame, the continuous-time state-space model follows

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{F}\mathbf{x}(t) + \mathbf{G}\mathbf{u}_{abc}(t) \quad (6.8a)$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t), \quad (6.8b)$$

with input vector

$$\mathbf{u}_{abc} = [u_a \ u_b \ u_c]^T \in \mathbb{Z}^3, \quad (6.9)$$

state-vector

$$\mathbf{x} = [\mathbf{i}_i^T \ \mathbf{v}_c^T \ \mathbf{i}_s^T \ \boldsymbol{\psi}_r^T]^T \in \mathbb{R}^8 \quad (6.10)$$

and output vector

$$\mathbf{y} = [\mathbf{i}_i^T \ \mathbf{v}_c^T \ \mathbf{i}_s^T]^T \in \mathbb{R}^6. \quad (6.11)$$

State-, input- and output matrices derived from (6.7) are

$$\mathbf{F} = \begin{bmatrix} \frac{-R_c - R_l}{L} \mathbf{I}_2 & -\frac{1}{L} \mathbf{I}_2 & \frac{R_c}{L} \mathbf{I}_2 & \mathbf{0}_2 \\ \frac{1}{C} \mathbf{I}_2 & \mathbf{0}_2 & -\frac{1}{C} & \mathbf{0}_2 \\ \frac{R_c L_r}{D} \mathbf{I}_2 & \frac{L_r}{D} \mathbf{I}_2 & -\left(\frac{1}{\tau_s} + \frac{R_c L_r}{D}\right) \mathbf{I}_2 & \left(\frac{1}{\tau_r} \mathbf{I}_2 - \omega_r \mathbf{J}\right) \frac{L_m}{D} \\ \mathbf{0}_2 & \mathbf{0}_2 & \frac{L_m}{\tau_r} \mathbf{I}_2 & -\frac{1}{\tau_r} \mathbf{I}_2 + \omega_r \mathbf{J} \end{bmatrix}, \quad (6.12)$$

$$\mathbf{G} = \left[ \frac{V_{dc}}{2L} \mathbf{I}_2 \quad \mathbf{0}_2 \quad \mathbf{0}_2 \quad \mathbf{0}_2 \right]^T \mathbf{K} \quad \text{and} \quad \mathbf{C} = [\mathbf{I}_6 \quad \mathbf{0}_{6 \times 2}]^T \quad (6.13)$$

with the diagonal matrices

$$\mathbf{I}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{J} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{0}_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}. \quad (6.14)$$

Converting the differential to the discrete time domain with sampling interval  $T_s$  results in the discrete-time state-space model, i.e. prediction model of the plant

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}_{abc}(k) \quad (6.15)$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k), \quad (6.16)$$

with

$$\mathbf{A} = e^{\mathbf{F}T_s}, \quad \mathbf{B} = -\mathbf{F}^{-1}(\mathbf{I}_8 - \mathbf{A})\mathbf{G} \quad \text{and} \quad \mathbf{C} = [\mathbf{I}_6 \quad \mathbf{0}_{6 \times 2}]^T. \quad (6.17)$$

## 6.5 FCS-MPC

### 6.5.1 Controller formulation

To adhere to the control objective stated above, the general cost function for FCS-MPC with reference tracking (2.13) is adopted as optimisation criterion for the plant. Hence, the relevant optimisation problem (2.36) with congruent matrices are considered. Tracking of the system outputs (6.11) requires weighing with the matrix  $\mathbf{\Lambda} \in \mathbb{R}^{6 \times 6}$  which is (by definition) positive semi-definite and symmetric. The system parameters sets the respective matrix dimensions to  $\mathbf{\Gamma} \in \mathbb{R}^{6N \times 8}$ ,  $\mathbf{\Upsilon} \in \mathbb{R}^{6N \times 3N}$ ,  $\mathbf{S} \in \mathbb{R}^{3N \times 3N}$ ,  $\mathbf{E} \in \mathbb{R}^{3N \times 3}$  and block diagonal matrix  $\tilde{\mathbf{\Lambda}} \in \mathbb{R}^{6N \times 6N}$ . Auxiliary matrices  $\mathbf{S}$ ,  $\mathbf{E}$  are constructed from diagonal matrices  $\mathbf{I}_3$  and  $\mathbf{0}_3$ . The input and reference sequences respectively are of length  $\tilde{\mathbf{u}}(k) \in \mathbb{R}^{3N}$  and  $\tilde{\mathbf{y}}^*(k) \in \mathbb{R}^{6N}$ .

### 6.5.2 Unconstrained minimum

Before optimisation can commence, the unconstrained minimum (2.33) must be computed, which in itself is a demanding task. In an attempt to alleviate the computational burden of  $\tilde{\mathbf{u}}_{unc}$ , consider  $\Theta$  from (2.26) in the expansion

$$\tilde{\mathbf{u}}_{unc}(k) = -\mathbf{Q}^{-1}((\mathbf{\Gamma}\mathbf{x}(k) - \tilde{\mathbf{y}}^*(k))^T \tilde{\mathbf{\Lambda}}\mathbf{\Upsilon} - \lambda_u(\mathbf{E}\mathbf{u}_{abc}(k-1))^T \mathbf{S})^T, \quad (6.18)$$

which can be rearranged to

$$\tilde{\mathbf{u}}_{unc}(k) = -((\tilde{\mathbf{\Lambda}}\mathbf{\Upsilon}\mathbf{Q}^{-T})^T \mathbf{\Gamma}\mathbf{x}(k) - (\tilde{\mathbf{\Lambda}}\mathbf{\Upsilon}\mathbf{Q}^{-T})^T \tilde{\mathbf{y}}^*(k) - \lambda_u(\mathbf{S}\mathbf{Q}^{-T})^T \mathbf{E}\mathbf{u}_{abc}(k-1))$$

and written in compact form

$$\tilde{\mathbf{u}}_{unc}(k) = \mathbf{V}\tilde{\mathbf{y}}^*(k) - \mathbf{W}\mathbf{x}(k) + \mathbf{Z}\mathbf{u}_{abc}(k-1), \quad (6.19)$$

with introduction of the matrices

$$\mathbf{V} = (\tilde{\mathbf{\Lambda}}\mathbf{\Upsilon}\mathbf{Q}^{-T})^T, \quad \mathbf{W} = \mathbf{V}\mathbf{\Gamma}, \quad \text{and} \quad \mathbf{Z} = \lambda_u(\mathbf{S}\mathbf{Q}^{-T})^T \mathbf{E}. \quad (6.20)$$

In this form (6.19), the online matrix computations can be reduced if  $\mathbf{V}$ ,  $\mathbf{W}$  and  $\mathbf{Z}$  are computed offline. Calculating  $\tilde{\mathbf{u}}_{unc}(k)$  also require the current state of the plant  $\mathbf{x}(k)$ , the sequence of output references  $\tilde{\mathbf{y}}^*(k)$  and the previous control  $\mathbf{u}_{abc}(k-1)$ . The latter is readily available from the previous sampling instant, but  $\mathbf{x}(k)$  and  $\tilde{\mathbf{y}}^*(k)$  not as much.

#### 6.5.2.1 Acquiring the state variables $\mathbf{x}(k)$

Acquisition of the first three variables in the state-vector (6.10) are obtained through direct measurement of the respective three-phase variables and converted to the stationary  $\alpha\beta$  coordinate system with the reduced Clarke transformation  $\mathbf{K}$  (2.38). Direct measurement of the rotor flux would require physical intrusion of the IM structure, hence the use of an open loop flux observer. Fundamentally, such an approach resembles real-time simulation of the rotor circuit equations [119]. Therefore we consider the rotor flux equation (2.65), formulated in the rotating  $dq$ -frame. Applying the forward Euler method thereto results in the corresponding discrete-time prediction

$$\boldsymbol{\psi}_r(k) = \begin{bmatrix} \psi_{rd}(k) \\ \psi_{rq}(k) \end{bmatrix} = \frac{L_m T_s}{\tau_r} \mathbf{i}_s(k-1) + \left(1 - \frac{T_s}{\tau_r}\right) \boldsymbol{\psi}_r(k-1). \quad (6.21)$$

Before insertion of  $\boldsymbol{\psi}_r(k)$  into the state-vector, conversion to the stationary  $\alpha\beta$  frame is required. It is accomplished with the transpose of the rotation matrix  $\mathbf{R}^T$  (2.40) to give

$$\boldsymbol{\psi}_r(k) = \begin{bmatrix} \psi_{r\alpha}(k) \\ \psi_{r\beta}(k) \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \psi_{rd}(k) \\ \psi_{rq}(k) \end{bmatrix}. \quad (6.22)$$

Note that the angle  $\theta$ , denotes the electrical angular position of the rotor. Once back in the  $\alpha\beta$  domain, the rotor flux angle is obtained with

$$\varphi(k) = \tan^{-1} \left( \frac{\psi_{r\alpha}(k)}{\psi_{r\beta}(k)} \right). \quad (6.23)$$

### 6.5.2.2 Acquiring the reference variables $\mathbf{Y}^*(k)$

The output reference sequence  $\tilde{\mathbf{y}}^*$  represents the evolution of the time-varying output reference vector  $\mathbf{y}^*$  over the prediction horizon. This implies that a unique reference vector is required for every step in the prediction horizon. To accomplish this, the drive system model is converted from the  $\alpha\beta$  frame to the rotating  $dq$  reference frame. The operating point, determined by the rotor flux setting, torque reference and the angular speed of the rotor allows for calculating the inverter voltage and subsequently, the output reference vector  $\mathbf{y}^*$ . This intricate process is well documented in [5, p.247] and a capitulation thereof is given in Appendix B.2. Figure 6.2 illustrates the relationship between the respective plant vectors, calculated for steady-state operation with the rated plant parameters. Note that the  $dq$ -frame rotates at an angular speed  $\omega_s$  and is aligned with the rotor flux vector  $\boldsymbol{\psi}_r$ . Once the plant vectors are computed, a final step requires the conversion of each component of  $\mathbf{y}^*$  from the  $dq$  reference frame back to the stationary  $\alpha\beta$ -frame for *every step* in the prediction horizon. The result is a predicted sequence reference vectors

$$\tilde{\mathbf{y}}^*(t) = \left[ \mathbf{y}^{*T}(t + 1T_s) \quad \mathbf{y}^{*T}(t + 2T_s) \dots \mathbf{y}^{*T}(t + NT_s) \right]^T. \quad (6.24)$$

### 6.5.3 Practical considerations of the MPC process

The fundamentals of MPC state that at every sampling instant ( $k$ ), the current state of the system  $\mathbf{x}(k)$  is captured and utilised as the initial state when predicting the future behaviour of the system  $\tilde{\mathbf{x}}(k)$  for a set of control variables  $\tilde{\mathbf{u}}(k)$ . The control variable that results in minimal cost is selected as the optimal control sequence  $\hat{\mathbf{u}}(k)$ , which contains the optimal control action  $\mathbf{u}_{abc}(k)$  for application to the plant.

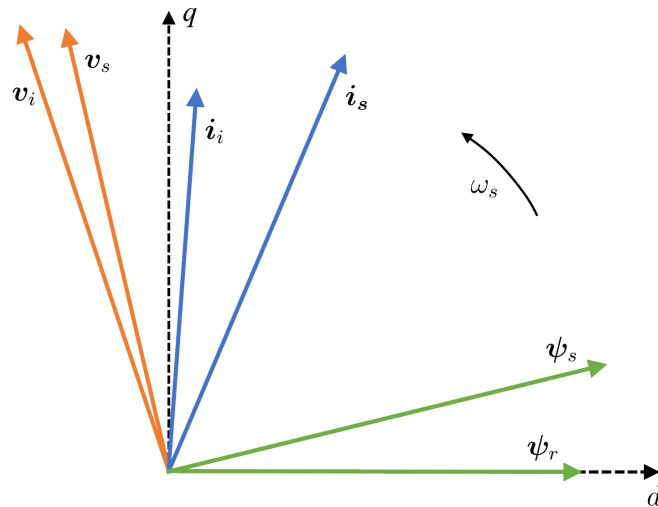


Figure 6.2: Vector quantities of the plant in the rotating  $dq$ -frame. Aligned with the rotor flux vector and rotating with  $\omega_s$ .

### 6.5.3.1 Delay compensation

A typical feature in simulated systems is that the entire MPC process is assumed to conclude in *zero time*. Figure 6.3 emphasises the idealistic process for a prediction horizon of  $N = 1$ . Note that the acquisition, computation and actuation processes occur instantaneously at the beginning of the sampling period ( $k$ ). Unfortunately delays are

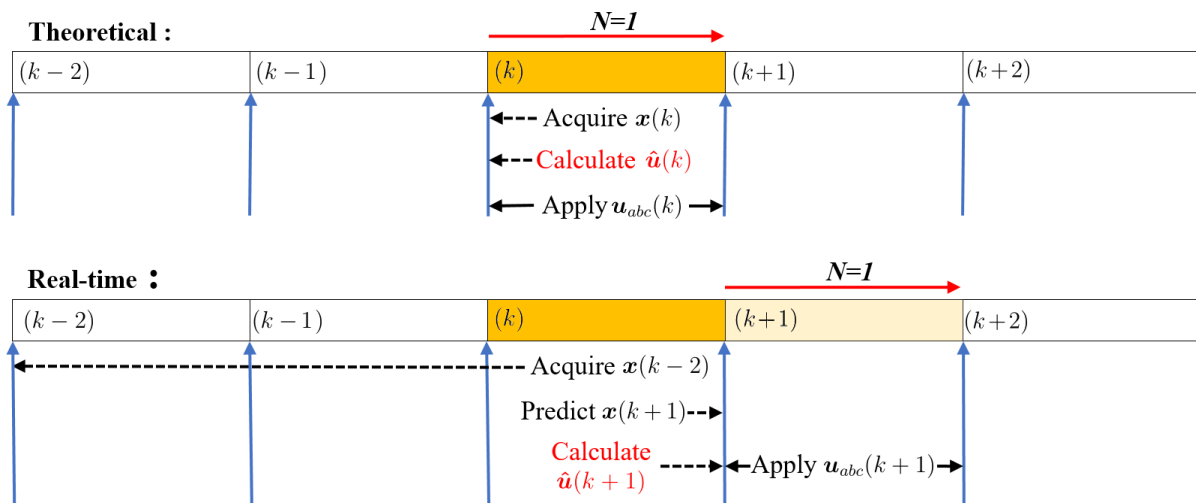


Figure 6.3: Illustration of the theoretical and real-time approach to MPC with prediction horizon  $N = 1$ .

unavoidable in real-world applications and must be compensated for. Direct measurement of the system state requires an analogue to digital converter (ADC), which introduces a combined acquisition and conversion delay. Added to this is a communication delay for



transferring the data to the computational unit where the predictive control algorithm is processed. After solving the MPC problem with its own computation delay, the optimum control action is produced. Before the plant can respond, another communication delay is experienced, together with an actuation delay caused by the converter driver circuits [120].

The real-time system used in this work effected an acquisition and communication delay of two sampling periods  $2T_s$ . Furthermore, the prediction model allows for switching transitions, only at the sampling instants. These limitations dictate that an optimal control can be computed during sampling period ( $k$ ), with system-state values acquired at sampling instant ( $k-2$ ). Once the control is computed, it is communicated to the plant for actuation at sampling instant ( $k+1$ ). Exposition of this real-time process is illustrated in Figure 6.3. In order to compute  $\hat{\mathbf{u}}(k+1)$ , the initial system state  $\mathbf{x}(k+1)$  and output reference  $\mathbf{y}^*(k+2)$  are required. Delay compensation is applied whereby the prediction model (6.15) are extrapolated from the known state  $\mathbf{x}(k-2)$  with the subsequent control actions resolve the initial system state

$$\mathbf{x}(k+1) = \mathbf{A}^3\mathbf{x}(k-2) + \mathbf{A}^2\mathbf{B}\mathbf{u}_{abc}(k-2) + \mathbf{A}\mathbf{B}\mathbf{u}_{abc}(k-1) + \mathbf{B}\mathbf{u}_{abc}(k). \quad (6.25)$$

The accuracy of this prediction is crucial to the performance of the MPC process [121]. We recall that the output reference sequence is time varying and conceived in the  $dq$ -frame with  $\varphi$  referencing the relative angular position of the rotating frame. Due to the delay in acquisition of the stator current  $\mathbf{i}_s$ , the response of the observer (6.21) is also delayed and result in  $\varphi(k-2)$  at sample period ( $k$ ). Moreover, the necessary adjustment to sample ( $k+1$ ) is

$$\varphi = \varphi + 3T_s\omega_s. \quad (6.26)$$

This advances the rotating  $dq$ -frame and accompanying reference output vectors forward in time from which, the sequence of output references are computed.

### 6.5.3.2 Rotor speed and look-up tables

Inclusion of  $\omega_r$  in the computation of the output reference sequence poses no serious complications. However, in the FCS-MPC process, the system-state matrix  $\mathbf{F}$  (6.12) and all the matrices computed therefrom, most importantly, the generator matrix  $\mathbf{H}$  are affected.

As a consequence, the demanding Cholesky decomposition (2.34b) must be executed. Following a brute force approach in computing  $\mathbf{H}$ , will drain computational resources and time, which in effect will inhibit the prediction horizon length. With current technology, this approach suffices for real-time MPC implementations with prediction horizons up to four or five, as demonstrated in the research of [95].

The research presented in this writing is concerned with facilitating longer prediction horizons, hence consideration of the alternative where some aspects are computed offline to facilitate a faster online process. In our case, it implies that for each of the affected matrices, a range of matrices should be computed as dictated by a predefined number of incremental speed steps. The *pu* speed range of the induction machine in this work is discretised into 1% steps, demanding an index of 100 matrices for each of the affected matrices. In light of this, the FCS-MPC process and other practical aspects presented above, necessitate preprocessing of the following matrices:

- system -state and -input matrices  $\mathbf{A}, \mathbf{B}$  for initial state prediction (6.25);
- matrices  $\mathbf{V}, \mathbf{W}$  and  $\mathbf{Z}$  for calculation of the unconstrained minimum (6.19);
- the matrix  $\mathbf{H}$ , for defining the CVP target  $\mathbf{H}\tilde{\mathbf{u}}_{unc}$  and computation of the initial sphere radius (3.48);
- matrix  $\mathbf{N}$  (3.17), for defining the lattice convex hull in the projection algorithm;
- matrix  $\mathbf{G}$  (5.7), to serve the SBDA in solving the CVP.

During the online process, and for every sampling period, the rotor speed  $\omega_r$  is discretised by the predictive controller to effect an index pointer

$$idx = \lfloor \omega_r \times 100 \rfloor, \quad (6.27)$$

which identifies the matrices relevant to the current speed. Note the additional speed input to the predictive controller (FCS-MPC block in Fig. 6.1).

#### 6.5.4 Controller settings

In order for a successful real-time implementation of long-horizon MPC, it is essential that the controller settings are carefully selected. Simulations of the proposed system

in MATLAB<sup>®</sup> assisted in the selection of the following parameters. All the forthcoming simulation results were obtained from simulating the system under steady-state conditions at the rated values stated in Table 6.1.

#### 6.5.4.1 Inverter switching frequency ( $f_{sw}$ )

The main objective of the controller is to ensure optimal reference tracking, while engaging the inverter to switch at a low frequency. This benefit of long-horizon MPC have reverberated throughout literature, and for the sake of continuity, we restate the findings of [5] by simulating the drive system at a sampling frequency of  $F_s = 8\text{kHz}$ . The relationship between the stator current total demand distortion  $I_{TDD}$ , and the switching frequency  $f_{sw}$  per semi-conductor device in a three-level NPC inverter is shown in Figure 6.4.

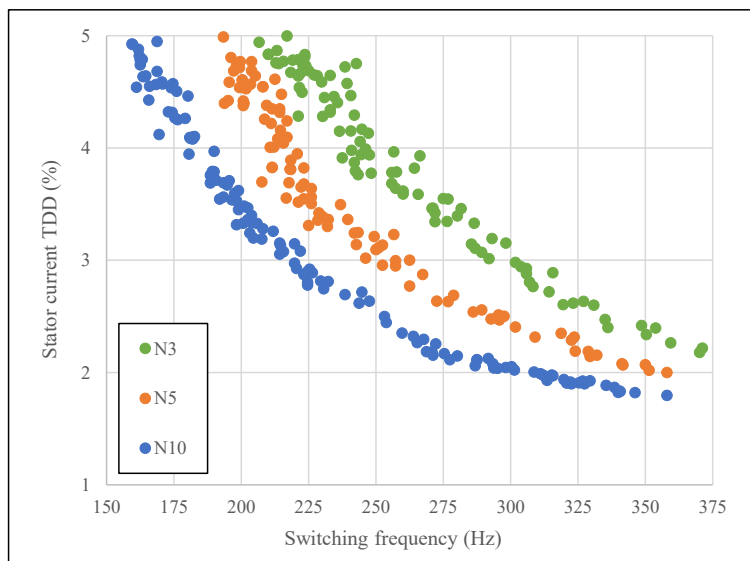


Figure 6.4: Effect of switching frequency on the stator current total dynamic distortion for selected prediction horizons  $N$ . The system sampling frequency is 8kHz.

From the simulated results it can be observed that an extended prediction horizon benefits lower switching frequencies more. Also, that higher switching frequencies yield a better performance in general. An additional complication to the control of this system is the resonant frequency  $f_{res}$  of the  $LC$  filter, which in this system equates to

$$f_{res} \approx \frac{1}{2\pi\sqrt{C\frac{LL_\sigma}{L+L_\sigma}}} = 303\text{Hz}, \quad (6.28)$$

where  $L_\sigma$  denotes the total leakage inductance of the IM. To avoid excitation of the filter resonance as far as possible in single-input single-output (SISO) systems, [122] and [123] state that  $f_{res}$  must be well above the fundamental frequency of the inverter output voltage, and well below the apparent switching frequency of the inverter ( $f_{inv}$ ), typically should  $f_{inv}/f_{res} > 2$ . This means for a three-level NPC inverter where the apparent switching frequency is twice the device switching frequency, a device switching frequency of approximately  $f_{sw} = f_{res}$  is required. However, the predictive control system presented here is a MIMO controller which has been shown in [5] to have the ability to lessen the ratio to  $f_{inv}/f_{res} < 1$ , provided, the prediction horizon is of significant length.

With consideration of the aspects mentioned above, and the objective of presenting a long-horizon predictive controller that can manage the system in *real time*, while delivering a relatively low stator current TDD, a device switching frequency of approximately 300Hz is selected.

#### 6.5.4.2 System sampling frequency

In [5] the *combination* of system sampling frequency and semiconductor device switching frequency was shown to have a significant impact on closed-loop performance of MPC. The selection of  $f_{sw}$  was based on the promise of a relative low sampling frequency, which dictates the sampling period  $T_s$  and grants the controller a certain amount of processing time before an output is expected. A low sampling frequency will increase the granularity of the sampled and control signals. Also, in a predictive controller, will the length of the *prediction horizon period* (6.1) increase. To investigate the influence of the sampling frequency  $F_s$  on the system, it was simulated for steady-state conditions at different sampling frequencies and various prediction horizons. In each of the simulations the switching weight  $\lambda_u$ , was adjusted to effect an average device switching frequency of  $f_{sw} = 300\text{Hz}$ .

The stator current TDD, representative of the drive performance, is plotted in Figure 6.5 relative to the sampling frequency, and in Figure 6.6 relative to the effected prediction horizon period. From Fig. 6.5 it can be noted that a definite minimum in terms of the stator current TDD exists for each of the prediction horizons at different sampling frequencies. The representation in Figure 6.6 furthers the notion, and indicates that a

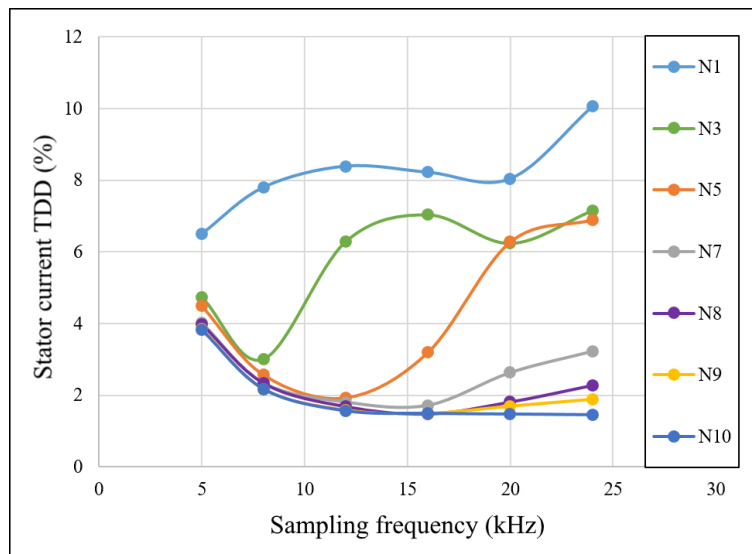


Figure 6.5: Effect of sampling frequency ( $F_s$ ) on the stator current total demand distortion ( $I_{s,TDD}$ ) for selected prediction horizons ( $N$ ).

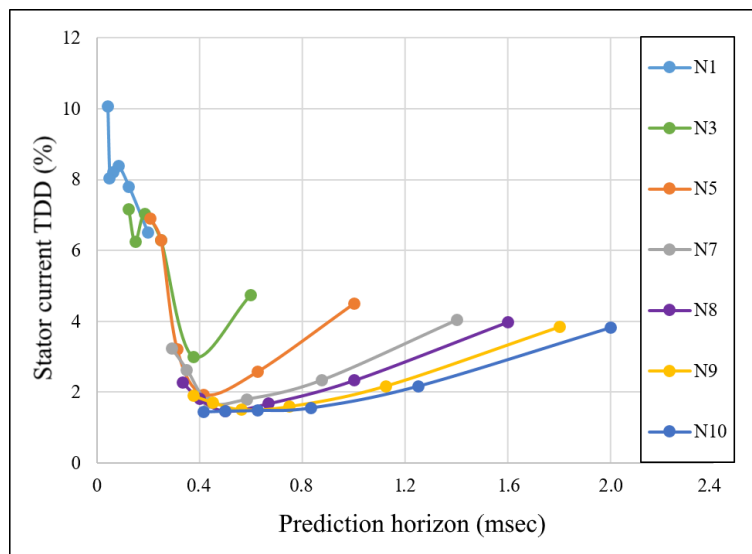


Figure 6.6: Relationship between the prediction horizon period ( $T_N$ ) and the stator current total demand distortion ( $I_{s,TDD}$ ) for selected prediction horizons ( $N$ ).

“prediction horizon period” should cover at least 45 degrees of the resonance period. It can also be noted that an extension of the prediction horizon period by means of a *reduced* sampling frequency, do not improve the stator current TDD. To conclude selection of the system sampling frequency, the computational burden of the sphere block decoding algorithm was also recorded. The worst-case scenario or maximum flop count, which occurs during *transients*, was considered and is shown in Figure 6.7 and Figure 6.8, respectively, in relation to the sampling frequency and prediction horizon period. Figure 6.7 conveys a general decrease in computational complexity, with an increase in sampling frequency.

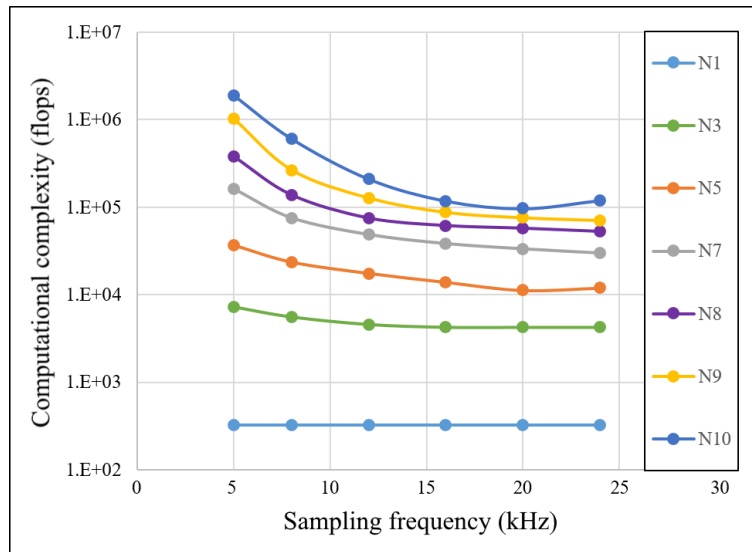


Figure 6.7: Effect of sampling frequency ( $F_s$ ) on the computational complexity (maximum flops) for selected prediction horizons ( $N$ ).

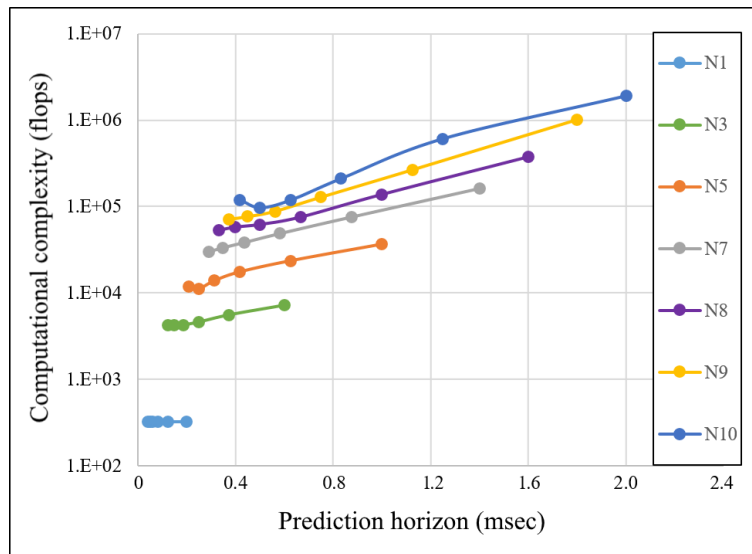


Figure 6.8: Relationship between the prediction horizon period ( $T_N$ ) and the computational complexity (maximum flops) for selected prediction horizons ( $N$ ).

The fact that longer prediction horizons increase computational cost is supported in both Figure 6.7 and Figure 6.8. In addition, Figure 6.8 shows that with an extension of the prediction horizon *period*, through reduction of the sampling frequency, the situation worsens. We recall from Section 2.5 that the transformation induced by a generator matrix results in a skewed lattice with skewness quantified in terms of the generator matrix's orthogonality defect (2.49). To this extent, Figure 6.9 presents the orthogonality defect of the generator matrix  $\mathbf{H}$  for the different prediction horizons at the various sampling frequencies. Interestingly, when comparing the orthogonality defects of Figure 6.9 with

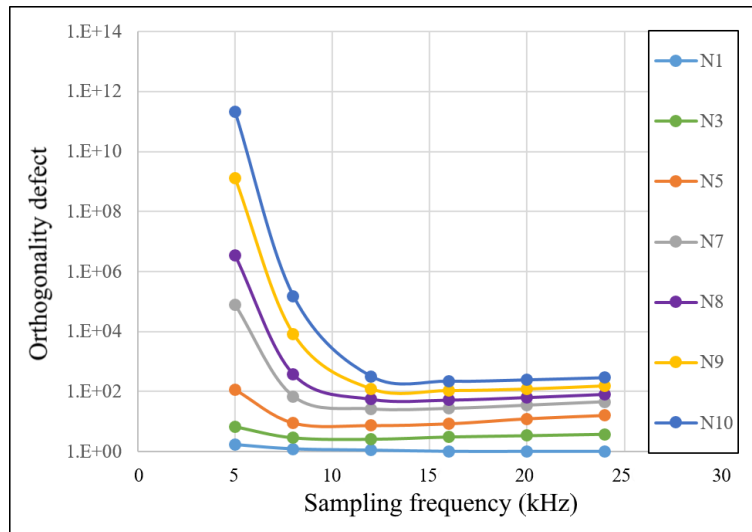


Figure 6.9: Orthogonality defect of the generator matrix  $\mathbf{H}$  for selected prediction horizons at various sampling frequencies.

the computational complexities of Figure 6.7, some correlation is evident. In [5, p.234] a similar observation was made in that a less orthogonal generator matrix increase the computational burden. Hence, the notion that an increased orthogonality defect predicts an elevated computational effort. In this system, the increased orthogonality defect influences the projection process described in subsection 3.6.4.1. To maintain optimality the projection hull is enlarged which directly affects the initial sphere-radius. Considering the observations made above, the system sampling frequency is selected to complement the switching frequency of 300Hz and as a compromise of the following:

- maximise the available computation time (low sampling frequency);
- minimise the computational effort (reduced orthogonality defect).

Trading some processing time for a reduction in computational complexity, especially when considering longer prediction horizons, identifies the most versatile sampling frequency of choice as 12kHz for prediction horizons 5 to 10. In summary, and for comparative purposes at a later stage, the simulated steady-state drive performances (MATLAB<sup>®</sup>) are listed in Table 6.2.

## 6.6 Experimental setup

Evaluation of the developed algorithms, i.e. the projection algorithm and sphere block decoding algorithm, were done in a real-time simulation (RTS) of the drive system described

Table 6.2: Idealistic performance metrics of the IM drive for steady-state operation (simulated in MATLAB<sup>®</sup>).

$N$	$F_s$ (Hz)	$f_{swd}$ (Hz)	$i_{i,TDD}$ (%)	$i_{s,TDD}$ (%)	$T_{e,TDD}$ (%)
<b>3</b>	8000	302	14.56	3.20	3.80
<b>5</b>	12000	303	13.06	2.11	2.32
<b>7</b>	12000	300	13.50	1.79	2.02
<b>8</b>	12000	300	13.57	1.76	2.01
<b>9</b>	12000	301	13.46	1.74	2.00
<b>10</b>	12000	300	13.35	1.61	1.82

above. The reasons being that this research is mainly concerned with developing an alternative decoding algorithm to the conventional SDA; and that such a complex MV drive system is costly and not readily available for random testing. Supporting the approach is the notion that real-time simulations are increasingly considered in high-quality scientific work, and are commonly published in conferences and transactions of well-known organisations such as IEEE or CIGRE. Industry also recognises and use the RTS approach as a standard method for the validation of engineering concepts [124, 125]. These simulators, operating under Windows and LINUX operating systems, avail the potential of integrating software tools such as SIMULINK for inter-system analysis.

### 6.6.1 Real-time simulation system

The OPAL RT-LAB system was selected for the real-time simulation and testing of the induction machine drive. The RT-LAB system consists out of two parts. One of which is a host computer that runs the RT-LAB software, and the other, the RT-LAB hardware or real-time simulator. Figure 6.10 [126] presents the fundamental layout of an OPAL RT-LAB system.

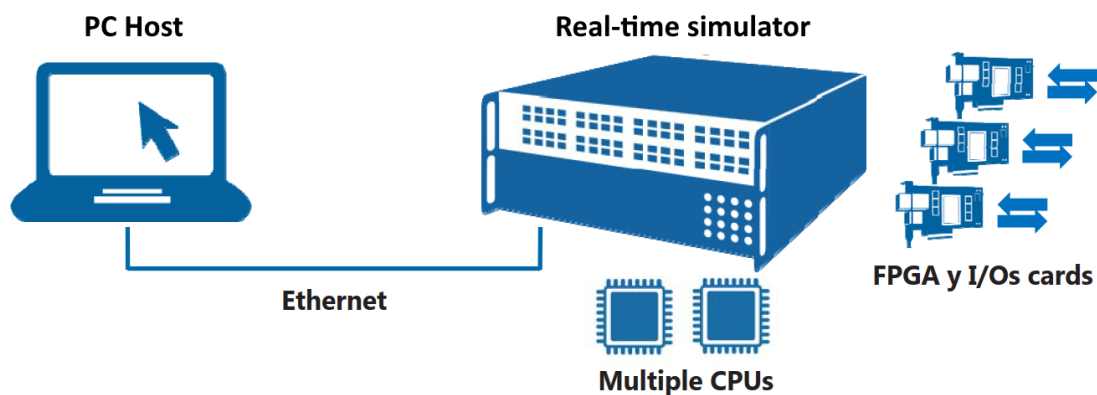


Figure 6.10: OPAL-RT technologies real-time simulation system.



The host computer,

- models the plant and control system in SIMULINK;
- compiles the model with RT-LAB software;
- and acts as the user interface during the simulation process.

Where, the RT-LAB simulator

- executes the real-time model;
- manages the input and output interfacing;
- and communicates with the host computer.

In this work, an INTEL<sup>©</sup> dual core<sup>20</sup> i5-6200U, 2.30GHz processor was employed as host computer with the real-time simulator an OPAL-RT technologies, model OP5600 platform. Architecturally, the simulator consists of a multi-core Xilinx 3.3GHz processor for simulating the real-time model, and a Spartan-3 FPGA for managing the analog and digital converter modules. Figure 6.11 [126] shows the device, and Figure 6.12 [126] the internal layout with LAN connection to the host computer.



Figure 6.11: OPAL-RT technologies, OP5600 platform.

### 6.6.1.1 Simulation process

Before running the real-time model *online* in the RT-LAB simulator, a number of steps needs to be followed *offline*.

<sup>20</sup>“Cores is a hardware term that describes the number of independent central processing units in a single computing component (die or chip). ”[127]

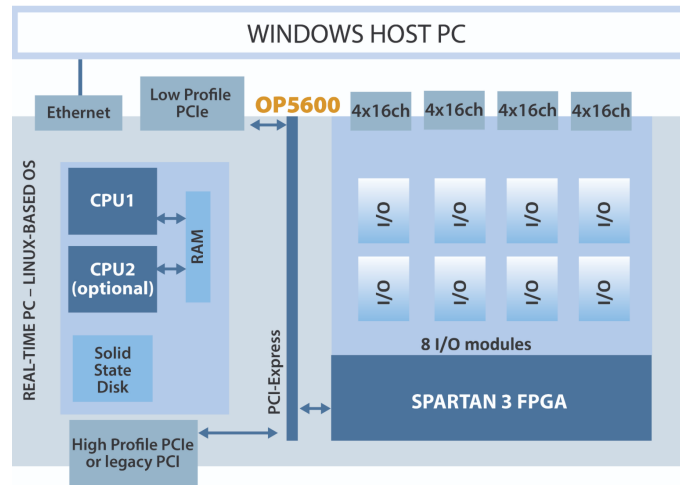


Figure 6.12: OP5600 internal layout.

- Firstly, an accurate model must be designed and successfully executed in the SIMULINK environment.
- Secondly, the model has to be grouped into three types of sub-systems, namely console, master and slave. The master and slave sub-systems contain the computational elements of the model whereas the console sub-system houses the user interface.
- Thirdly, the SIMULINK model must be converted to C-code with the RT-LAB software.

Before online simulation can commence, the master and slave sub-systems are selected for download to different cores of the real-time simulator. Upon completion of the C-code download, the model is executed. The user interface, i.e. console sub-system on the host computer, allows for manipulating system input parameters and also gives access to visualisation of the system responses in real time.

### 6.6.2 Drive system realisation

Realisation of the drive system in the RT-LAB system is illustrated in Figure 6.13. Note that the controller and plant are implemented on separate CPU cores and interfaced via the FPGA controlled analogue and digital IOs with a hardwired external loop. The external loop enforces hardware synchronisation, and in the process solidifies the performance results of the *controller* in real time. The IM speed  $\omega_r^*$  and flux  $\Psi_r^*$  reference settings, together with the mechanical load torque  $T_l$ , are managed from the host com-

puter. Measurements of the plant during test procedures are recorded in the RT simulator and transferred to the host upon completion.

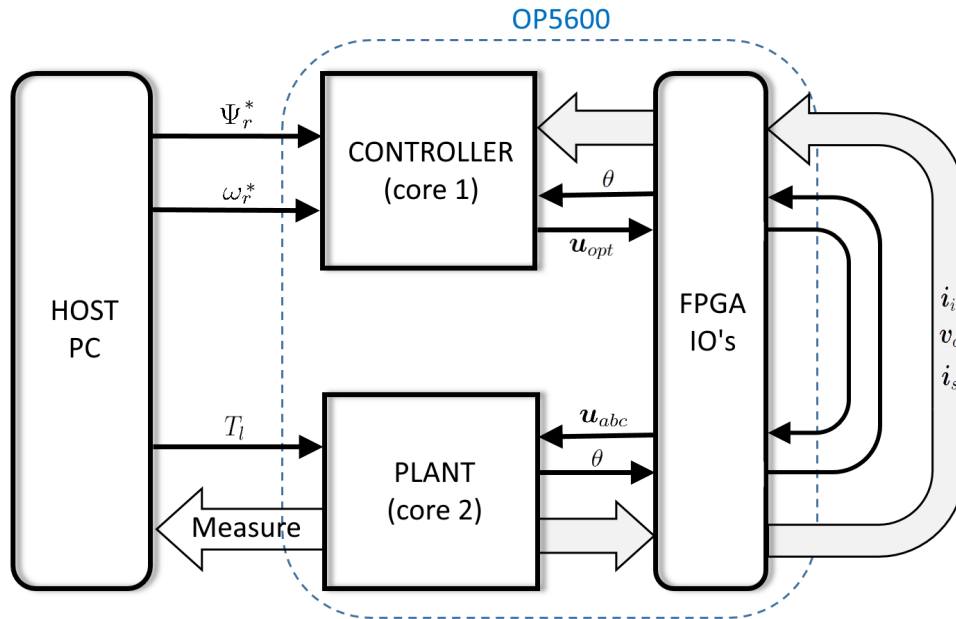


Figure 6.13: Experimental setup.

Modelling of the plant, i.e. the inverter, the  $LC$ -filter and squirrel-cage induction motor with an appropriate mechanical load was realised with OPAL-RT's state-space nodal (SSN) components. The SSN method achieves a fast and accurate simulated response without introducing artificial delays. Upon actuation of the plant with  $\mathbf{u}_{abc}$ , a corresponding response of the output variables was available within the same sampling period and communicated to the FPGA IOs. Converting the plant states to an analogue value and sampling it back to a digital counterpart at the controller, resulted in the  $2T_s$  delay, which was discussed in subsection 6.5.3.1. For a more detailed expansion of the drive system, modelled in SIMULINK, the reader is referred to Appendix C.

To demonstrate the performance of the predictive controller and supporting algorithms, the drive was subjected to steady-state and transient conditions. In all instances, the more demanding *constant torque load*<sup>21</sup> was considered, rather than the more common *variable torque load*<sup>22</sup>.

<sup>21</sup>A constant torque load demands the same amount of torque throughout the speed range while the horsepower changes linearly with speed.

<sup>22</sup>A variable torque load or quadratic torque load demands less torque at speeds below the base speed. The torque varies as the square of the speed and the horsepower varies as the cube of the speed.

## 6.7 Steady-state performance

Real-time simulation of the drive was performed for steady-state conditions with a constant torque load of  $T_l = 1\text{pu}$  at the rated speed  $\omega_r^* = 0.9911\text{pu}$  and optimal machine magnetisation  $\Psi_r^* = 0.9117\text{pu}$ . The parameters listed in Table 6.1 were considered with the dc-link voltage stiff and the neutral point voltage of the NPC inverter fixed at zero. To obtain efficient tracking of the stator current, the main diagonal of the weighing matrix  $\Lambda$  was populated with the vector

$$[\lambda_{i_i,\alpha}, \lambda_{i_i,\beta}, \lambda_{v_c,\alpha}, \lambda_{v_c,\beta}, \lambda_{i_s,\alpha}, \lambda_{i_s,\beta}] = [1, 1, 5, 5, 100, 100]. \quad (6.29)$$

The switching weight  $\lambda_u$ , was adjusted to effect an average device switching frequency  $f_{sw} = 300\text{Hz}$ . Visually, the resulting waveforms can be observed in Figure 6.14 for the prediction horizon  $N = 8$  case. Measurements of the inverter current  $\mathbf{i}_i$  and stator current  $\mathbf{i}_s$  were employed to compute their respective total demand distortions, i.e.  $i_{i,TDD}$  and  $i_{s,TDD}$ . The torque total demand distortion  $T_{e,TDD}$  was obtained by calculating  $T_e$  as per (2.67) from the measured stator current  $\mathbf{i}_s$  and estimated rotor flux  $\psi_r$ . Table 6.3 list the results for prediction horizons  $N = 3, 5$  and  $8$ . At first glance, the dramatic

Table 6.3: Performance metrics of the drive system during steady-state operation (OPAL real-time simulation).

$N$	$F_s$ (Hz)	$f_{swd}$ (Hz)	$i_{i,TDD}$ (%)	$i_{s,TDD}$ (%)	$T_{e,TDD}$ (%)
<b>3</b>	8000	314.3	15.13	3.59	4.10
<b>5</b>	12000	301.3	13.37	2.54	2.59
<b>8</b>	12000	307.8	13.83	1.87	2.32

effect of the  $LC$  filter on the current TDD can be noticed. The input current to the filter  $\mathbf{i}_i$ , with a fairly high TDD, is transformed to result in the stator current  $\mathbf{i}_s$  with a low single-digit value. As a drive performance indicator, this clearly improves with extension of the prediction horizon. Compared to the theoretical MATLAB<sup>®</sup> simulations (Table 6.2), the real-time results show a general decrease in the drive performance. Considering that the real-time simulation utilises a simulated plant, accurate responses are expected. However, inaccuracies and the delays instilled by the analogue-to-digital and digital-to-analogue conversion processes, influence the measurements and the delay compensation process. Both crucial in obtaining the *initial* state variables of the model. This accounts

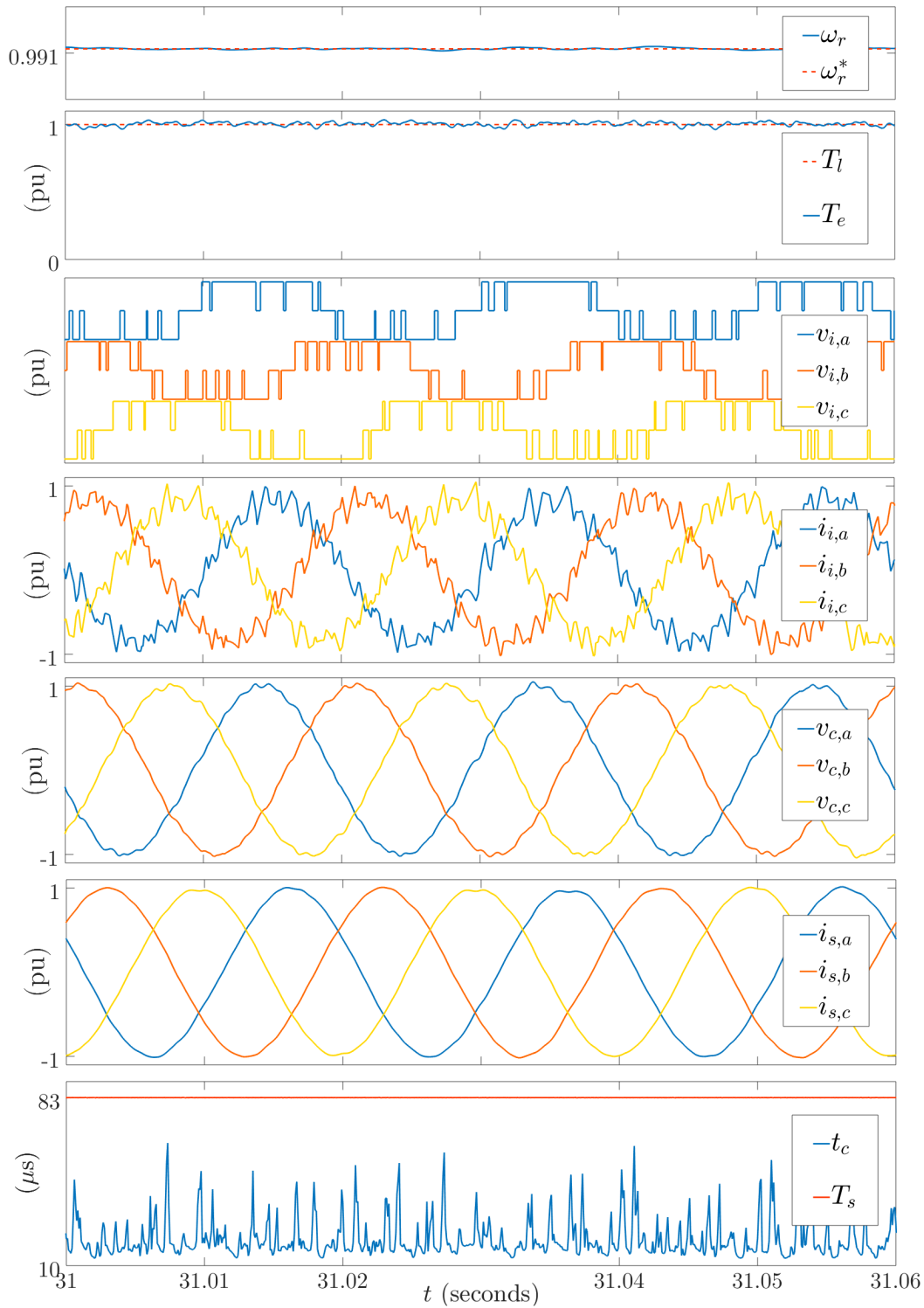


Figure 6.14: System waveforms for steady-state operation of the three-level inverter drive at nominal speed and torque. Specific to the  $N = 8$  case with an average device switching frequency of 300Hz. Rotor speed  $\omega_r$ , developed torque  $T_e$ , inverter voltage  $\mathbf{v}_i$ , inverter current  $\mathbf{i}_i$ , capacitor voltage  $\mathbf{v}_c$ , stator current  $\mathbf{i}_s$  and controller computation time  $t_c$ .

for the deviation from the idealistic MATLAB<sup>©</sup> results.

Performance of the controller algorithm, was quantified in terms of its computation time  $t_c$ . Reported by a function of the OPAL-RT software,  $t_c$  is plotted in the last graph of Figure 6.14. Note that due to the sampling frequency of 12kHz, the computation time is bounded to  $83\mu\text{s}$ . Under steady-state operation, and for prediction horizons 3,5 and 8, the control problem was readily solved in real time with no *overruns*<sup>23</sup> detected. However, prediction horizons 9 and 10 demanded too big a computational effort for successful termination of the controller algorithm.

## 6.8 Transient performance

Response of the drive to transients, was evaluated by considering the respective effects that a load torque step and speed reference step have on the IM drive. In both cases, the longer prediction horizon  $N = 8$  controller was selected with the weighing matrix  $\Lambda$  augmented with (6.29). The respective switching weights used for steady-state conditions were again selected for the respective prediction horizons.

### 6.8.1 Load torque step

The IM is set to run at the nominal speed ( $\omega_r = 0.9911\text{pu}$ ) with no load connected ( $T_l = 0\text{pu}$ ). A constant torque load presenting a torque of  $T_l = 0.8\text{pu}$  is connected at an instant, demanding corrective measures from the controller. Figure 6.15 shows the abrupt step in load torque, with the subsequent torque response  $T_e$  and motor speed  $\omega_r$  variation. The other waveforms respectively depict the system outputs and the controller computation time during this dynamic period. From the speed response, it can be observed that the *transient deviation* from the speed reference is in the order of  $0.991 - 0.985 = 0.006\text{pu}$ , and the *transient response time* approaches 40ms before settling in the steady-state regulation band. The torque response  $T_e$  of the IM increased from zero to 0.8pu in  $\pm 8\text{ms}$ . This relative slow response is commanded by the PI speed controller due to the small deviation in speed ( $\omega_r - \omega_r^*$ ). The result is a smooth transition from zero to maximum torque without overshooting 1pu. This is also noticeable in the system waveforms where the majority of variables are maintained within their rated values. The capacitor voltage however, shows

---

<sup>23</sup>An “overrun” occurs if the operations of the real-time simulator are not achieved within a period of the selected *fixed time-step*, i.e. sampling period. In such a case the real-time simulation is considered erroneous.

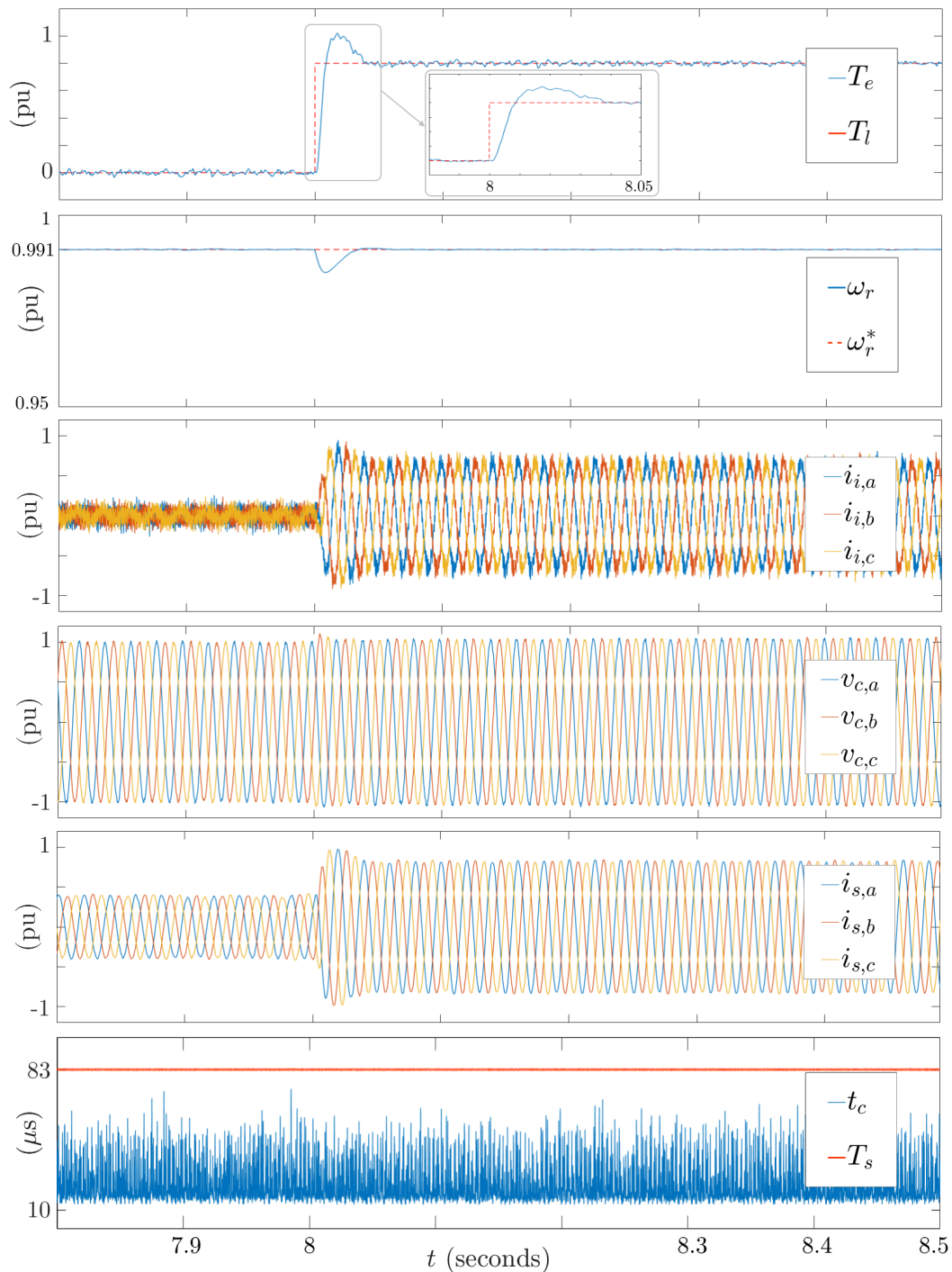


Figure 6.15: System waveforms corresponding to a variation in the mechanical load from 0pu to 0.8pu. Developed torque  $T_e$ , rotor speed  $\omega_r$ , inverter current  $i_i$ , capacitor voltage  $v_c$ , stator current  $i_s$  and controller computation time  $t_c$ .

a slight transient at the onset of the torque step that marginally exceeds 1pu. In the last graph of Figure 6.15, the computational time of the controller remains well below the sampling period of  $83\mu\text{s}$ . No real increase in computational complexity can be noticed with the onset and completion of the torque transition period. This is mainly due to the effect of target preconditioning provided by the projection algorithm.

### 6.8.2 Speed reference step

A constant torque load of 0.8pu was selected for evaluating the drive performance during acceleration of the IM. The main reason for choosing an 80% loading factor is to preserve the inverter and system ratings when a rapid increase in motor speed is demanded. To test the drive under such duress, the motor was loaded with  $T_l = 0.8\text{pu}$  and driven at a speed of  $\omega_r = 0.2\text{pu}$ . At an instant, the speed reference was stepped up to 1pu. Such a command effects a large transient deviation ( $\omega_r - \omega_r^*$ ), which subsequently leads to the PI speed controller issuing a maximum torque request. Figure 6.16 shows the step in the speed reference, together with the rotor speed  $\omega_r$ , delayed by the combination of system inertia and available torque. An almost immediate torque response  $T_e$  can be observed, which is maintained at a maximum until the required speed is attained. From the speed curve, a *rise time* of 1.83s and a *settling time* of 1.86s can be concluded, which sets the bandwidth of the drive to 0.54Hz. The enlarged view of the machine torque exhibits the rapid torque adjustment made by the predictive controller; a settling time of less than 10ms is achieved with minimal overshoot. During the acceleration process, the inverter current is slightly elevated while the stator voltage increase with the speed. The stator current is optimally maintained during the transient period, as its tracking weight (6.29) is much larger than the weights assigned to the inverter current and capacitor voltage. As mentioned earlier, these priorities can be altered to suit specific drive design requirements.

The computational time required by the predictive controller shows an almost negligible increase during the transient period in comparison to the steady-state conditions. Again, this containment of the computational burden during transient occurrences is attributed to preconditioning of the target.

### 6.8.3 Computational benchmark

As a comparative measure and to benchmark the performance of the SBDA, the predictive controller was equipped with the conventional SDA. The SDA was provided with the advantage of target preconditioning. A prediction horizon of  $N = 8$ , a sampling frequency of 12kHz and weighing as specified in section 6.7, were selected. Subjecting the drive to steady-state conditions resulted in the computation times shown in Figure 6.17. From the results it can be noted that the computation time, in general, is significantly higher than



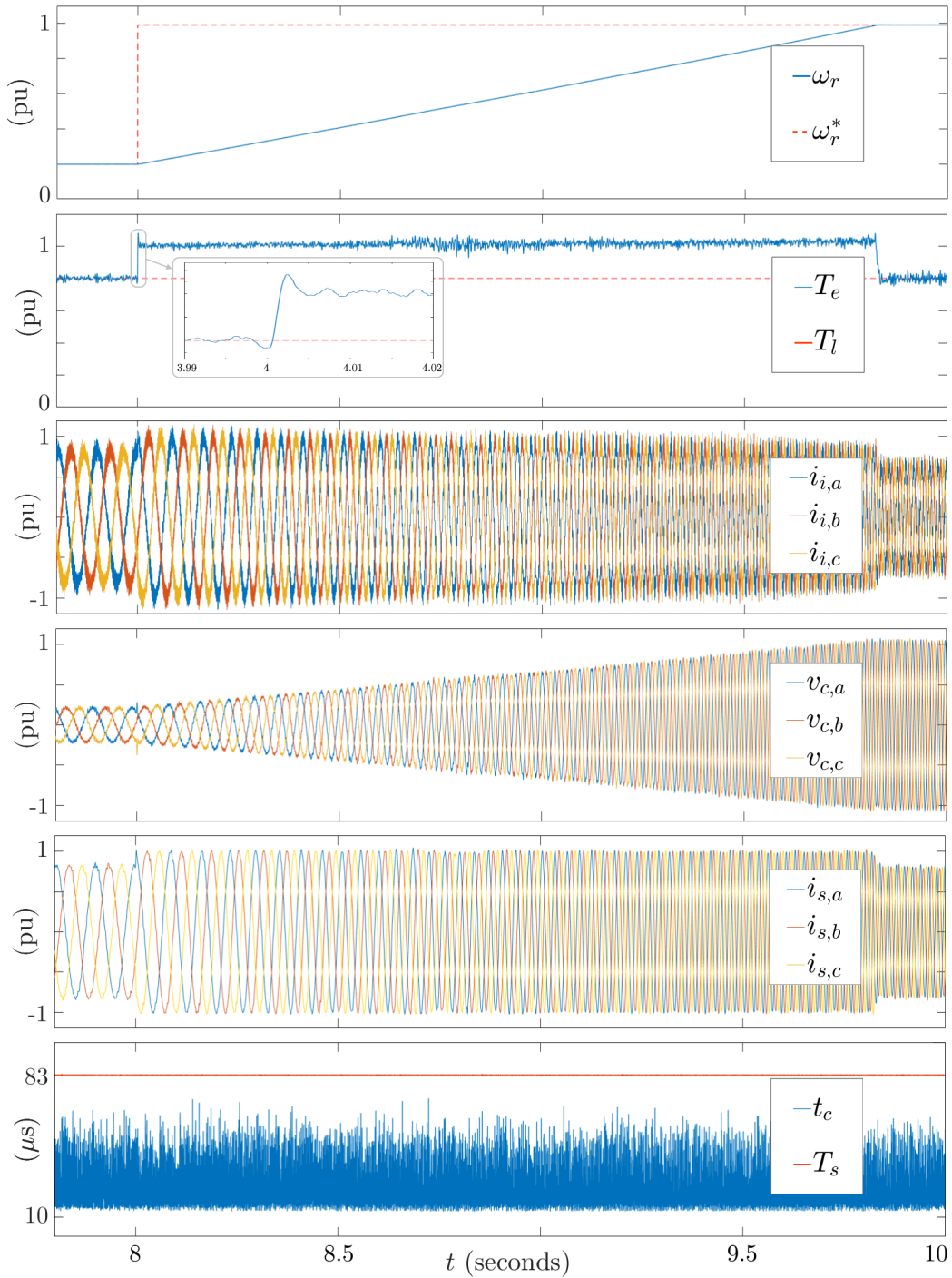


Figure 6.16: System waveforms corresponding to a speed reference step from 0.2 pu to 1 pu. Rotor speed  $\omega_r$ , developed torque  $T_e$ , inverter current  $\mathbf{i}_i$ , capacitor voltage  $\mathbf{v}_c$ , stator current  $\mathbf{i}_s$  and controller computation time  $t_c$ .

the time recorded when employing the SBDA (Fig. 6.14). Even though the operational conditions were favourable, i.e. steady-state, the SDA struggled to solve the CVP in time, and frequently exceeded the sample-period of  $83\mu\text{s}$ . Due to this, testing under more challenging conditions (transients) was omitted.

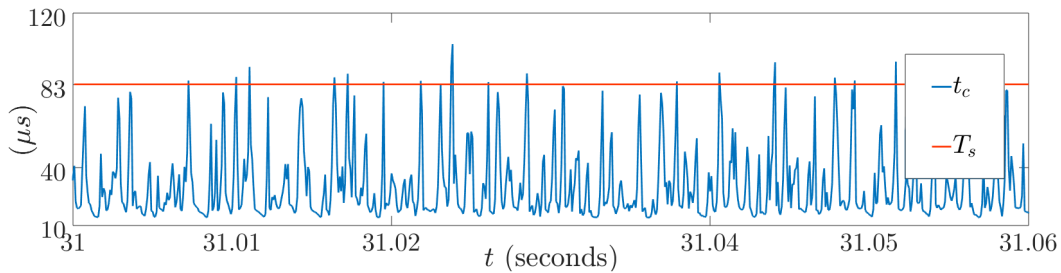


Figure 6.17: SDA computation time for solving the  $N = 8$  MPC problem.

## 6.9 Increased voltage levels

In subsection 5.4 the impact of multiple inverter voltage levels on the modelling of an FCS-MPC system and the resulting CVP was presented. MATLAB<sup>®</sup> simulations of the relevant power electronic system, suggested that the Micciancio Voulgaris algorithm can be considered as an alternative solver, especially for CVPs in lattices with an increased structure (page 102-104). To investigate the performance of the MVA as solver in real time, the NPC inverter in the drive system presented above was replaced with a cascaded H-bridge of different voltage levels.

### 6.9.1 Experimental results

A variation in the number of inverter voltage levels  $l_v$  necessitates the redefinition of the three-phase inverter output voltage  $\mathbf{v}_i$  (6.3) and control set  $\mathbf{U} = \mathcal{U}^3$ , respectively as per (2.43) and (2.44). Also, the dc supply ( $v_{dc}$ ) per CHB cell is adjusted so that for a  $l_v$ -level CHB inverter the sum of all dc supplies equate to the dc supply  $V_{dc}$  parameter listed in Table 6.1, i.e.

$$V_{dc} = v_{dc}(l_v - 1). \quad (6.30)$$

For steady-state conditions the adapted drive was simulated in real time for multilevel inverters with voltage levels  $l_v = \{3, 5, \dots, 11\}$ . To limit the complexity of the basic Voronoi cell (3.38), a MPC prediction horizon of  $N = 3$  is selected. The 12kHz sampling frequency was again chosen with the switching weight  $\lambda_u$ , adjusted to effect an apparent inverter switching frequency of 600Hz. As a reference, this translates to an average device switching frequency of 300Hz for a three-level inverter (2.50, 2.51). Figure 6.18 show the waveforms resulting from the FCS-MPC drive with a five-level inverter topology.

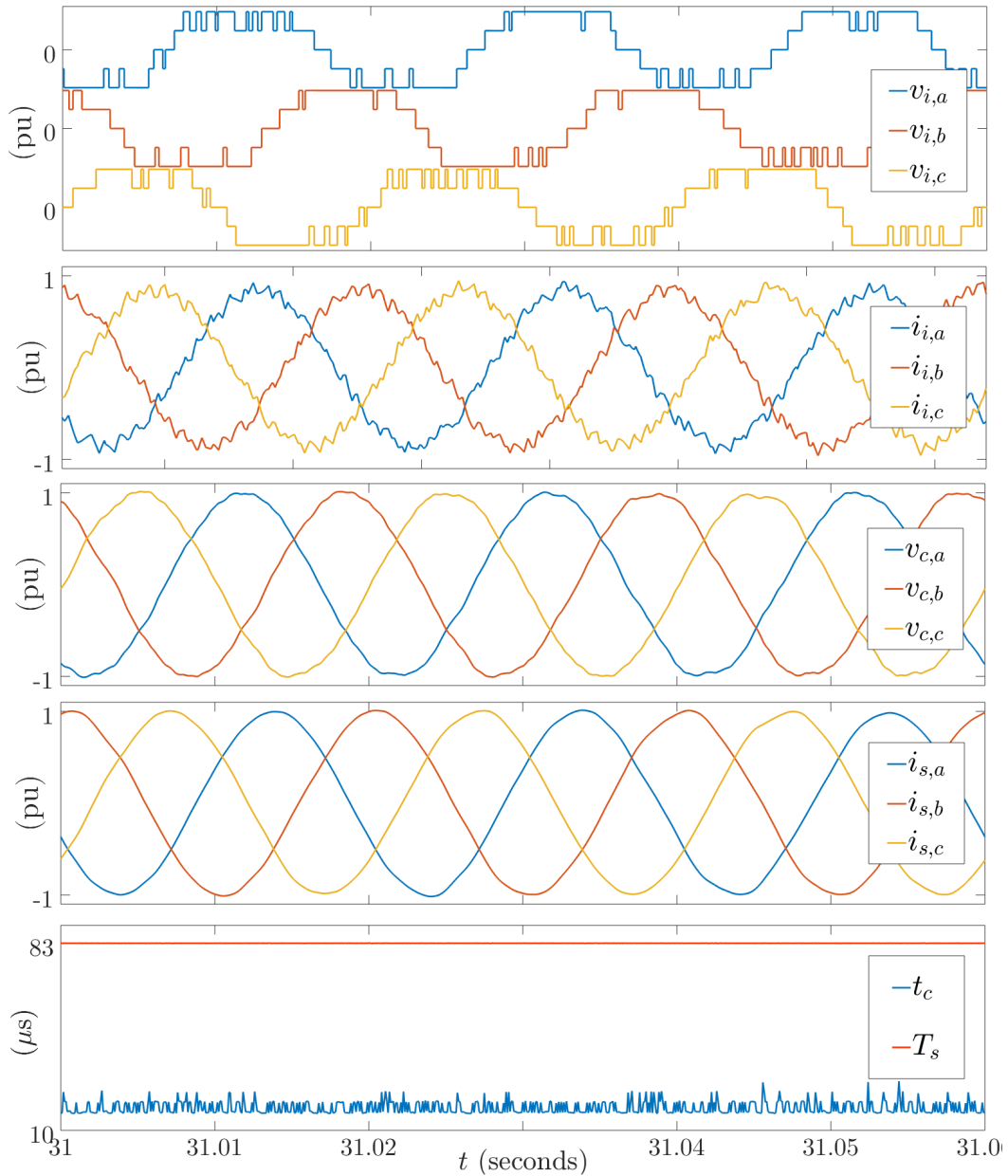


Figure 6.18: System waveforms for steady state operation of the five-level CHB inverter drive at nominal speed and torque. Specific to the  $N = 3$  case with an average device switching frequency of 150Hz. Rotor speed  $\omega_r$ , developed torque  $T_e$ , inverter voltage  $v_i$ , inverter current  $i_i$ , capacitor voltage  $v_c$ , stator current  $i_s$  and controller computation time  $t_c$ .

A visual comparison of the waveforms in Figure 6.18 to those of the three-level NPC inverter with prediction horizon  $N = 8$  (Fig. 6.14), suggests an improvement in the inverter current TDD with seemingly similar results for both stator current TDDs. These aspects can also be noticed in the real-time results listed in Table 6.4 for the various inverter topologies. As a first observation, the improvement in stator current TDD can

Table 6.4: Performance metrics of the drive system during steady-state operation (OPAL real-time simulation).

$N$	$F_s$ (Hz)	$l_v$	$f_{swd}$ (Hz)	$\mathbf{i}_{i,TDD}$ (%)	$\mathbf{i}_{s,TDD}$ (%)	$T_{e,TDD}$ (%)	$t_{c,max}$ ( $\mu$ s)
<b>3</b>	12000	3	304	14.02	8.45	8.16	33.38
<b>3</b>	12000	5	152	9.93	3.86	4.18	33.28
<b>3</b>	12000	7	100	8.58	2.45	3.07	35.90
<b>3</b>	12000	9	77	7.75	1.99	2.90	38.53
<b>3</b>	12000	11	62	6.91	1.95	2.82	38.94

be noticed when the inverter levels are increased from 3 to 5. It must be mentioned that the magnitude of the improvement is due to the 12kHz sampling frequency which is *non-optimal* for the three-level inverter with prediction horizon  $N = 3$ . This can be noted from Table 6.3 where the performance of this combination with a sampling frequency of 8000Hz results in a stator current TDD of approximately 3.6%. However, the higher sampling frequency improved the inverter current TDD. The higher number of inverter voltage-levels recorded a gradual decrease in stator current TDD to where the eleven-level topology, with MPC prediction horizon  $N = 3$ , almost matched the three-level topology, with MPC prediction horizon  $N = 8$ . In addition, this is accomplished at a fifth of the device switching frequency. More significant is the improvement in inverter-current TDD, which is almost half for the eleven-level topology compared to the three-level topology. In comparison, Table 6.5 lists the corresponding theoretical results.

Table 6.5: Performance metrics of the drive system during steady-state operation (MATLAB<sup>©</sup> simulation).

$N$	$F_s$ (Hz)	$l_v$	$f_{swd}$ (Hz)	$\mathbf{i}_{i,TDD}$ (%)	$\mathbf{i}_{s,TDD}$ (%)	$T_{e,TDD}$ (%)
<b>3</b>	12000	3	302	13.35	7.05	7.67
<b>3</b>	12000	5	150	7.17	3.37	3.98
<b>3</b>	12000	7	100	4.69	2.06	2.5
<b>3</b>	12000	9	75	3.85	2.01	2.48
<b>3</b>	12000	11	60	3.51	1.88	2.38

## 6.9.2 Computational performance

Performance of the predictive controller in terms of algorithm termination time is shown in the last graph of Figure 6.18. Table 6.4 also lists the maximum termination times for all the multilevel inverters tested. Relatively low termination times were recorded with the MVA as solver, and as expected, only a marginal increase manifested as the

number of inverter levels  $l_v$  increased. Comparative performances of the conventional SDA and proposed SBDA in the same system and under the same conditions were also obtained. The SBDA performed poorly, mainly due to the increased size of the inverter control set  $\mathcal{U}$ . From section 5.1 we recall that the search tree of the SBDA branch into  $l_v^3$  three-dimensional control vectors to constitute a search tree of depth  $N$ . In the three-level case, a set of 27 control vectors resulted, which were manageable and suited the processing hardware well. Moreover, the five-level inverter case constitutes 125 control vectors which in the decoding process demand  $(3 \times 125)$ -matrix computations at every node visit. This burden proved to be excessive, as the SBDA struggled to find the optimal solution within the allocated sample period of  $83\mu\text{s}$ . Performance simulations of the conventional SDA in Section 5.4 suggested that poor termination times can be expected (Fig. 5.7). On the contrary, the SDA recorded similar termination times to the MVA on the real-time platform.

## 6.10 Summary

In this chapter the proposed projection algorithm and alternative sphere block decoding algorithm (SBDA) were evaluated in an FCS-MPC scheme for an IM drive. The prospect of utilising the Micciancio Voulgaris algorithm (MVA) as decoder to the MPC problem for multilevel inverter topologies was also evaluated in the same drive system. Equipped with target preconditioning from the projection algorithm, the SBDA outperformed the conventional SDA in solving the FCS-MPC problem relevant to a three-level inverter. Since, the OPAL-RT real-time simulator availed only one CPU core to the controlling algorithms, the performance of the SBDA is mainly attributed to the offline precomputation of matrices. The SBDA, which was proposed in specific for the three-phase, three-level NPC inverter, struggled with solving the MPC problem for inverter topologies with an increased number of voltage levels. This is as a result of the increased size of the inverter control set. The conventional SDA however, outperformed the SBDA in these instances and recorded termination times similar to the MVA. Due to the relation of the basic Voronoi cell complexity to the MPC prediction horizon, a horizon of only  $N = 3$  could successfully be implemented in real time. However, the consistency of the MVA's termination times supported its deterministic nature and encourages the consideration of FCS-MPC problems which result in lattices with a larger structure.

# Chapter 7

## Conclusions

In this thesis, the FCS-MPC control technique for power electronic systems of higher order was shown to be executable in real time. These types of systems require relatively long prediction horizons which are hard to achieve with the conventional SDA. To address this well-known issue, alternative decoding strategies were developed, with special attention given to the type noted as “exponential space algorithms”. Testing of the renowned MVA in the FCS-MPC of a typical power electronic application exposed its limitations, but also revealed some attractive traits in terms of computability. Some of these traits, favouring current processing technology in particular, were incorporated into the SDA of [39].

### 7.1 Contributions

The primary objective was achieved by delivering a faster decoding algorithm that allowed for extending the length of the prediction horizon of FCS-MPC for a three-phase system up to eight steps. Secondary contributions include a fast projection algorithm with improved optimality and the matching of the Micciancio Voulgaris algorithm to a suitable power electronic application. The research process followed in this study led to some original contributions, which are explained below per chapter.

#### 7.1.1 Chapter 3

In this chapter, the lattice search space resulting from the underlying FCS-MPC problem was presented. Partitioning of the space into a Voronoi diagram gave insight into the complexity of high-dimensional lattice structures and the decoding process followed by

the Micciancio Voulgaris algorithm. Considering the MVA as solver to the CVP of a truncated lattice required projection onto the convex hull of the lattice. This led to the development of a projection algorithm which is tailored to the lattice structures associated with the FCS-MPC of multilevel inverters. The algorithm employs aspects of the MVA and similarly guarantees a solution within a fixed number of algorithm iterations. Space partitioning of a truncated lattice into a Voronoi diagram revealed the sub-optimality experienced by previous preconditioning approaches, and assisted in the adaption of the proposed projection algorithm to deliver in improved optimality.

### 7.1.2 Chapter 4

Performance of the various decoding algorithms was simulated in this chapter. A relatively simple PES was considered as the plant, since the main objective was the performance of the respective algorithms. Through these simulations, it was shown that the efficiency of an algorithm relies on more factors to consider than its computational complexity (flop count) alone. The exiting prospect of the MVA as a single exponential time solver, faded as its exponential space complexity proved to be too demanding for solving high dimensional CVPs. However, the observation that the MVA, with the highest computational complexity, terminated faster than the other algorithms for lower dimensional CVPs contributed to the notion that modern processors favour matrix processing. The proposed projection algorithm's efficiency was demonstrated through comparison with the MATLAB<sup>®</sup> interior-point-method and the Multi-Parametric toolbox, distance-function. Due to its tailored design, the proposed algorithm performed well above the multi-purpose commercial solvers.

### 7.1.3 Chapter 5

In this chapter, the sphere decoding algorithm was modified by introducing block-matrix computations. This increased the computational complexity of the proposed algorithm, but also sanctioned the use of preprocessing. Partitioning of the generator matrix into manageable blocks allowed for online block-matrix processing and efficient precomputation and storage offline. This reduced the computational complexity of the proposed approach, and through simulation showed that good algorithm termination times can be expected from the Sphere Block Decoding algorithm. Although the MVA was discarded as

a practical solver to CVPs in high-dimensional lattices, its deterministic properties were shown to suit larger lattice structures. Lattices typically emerge from the FCS-MPC of multilevel inverters with an increased number of voltage levels.

#### 7.1.4 Chapter 6

Hardware in the loop, real-time testing of the proposed algorithms in an FCS-MPC drive application was done in this chapter. The emulated IM drive was a third-order system which is relatively difficult to control. Assisted by the proposed SBDA, the FCS-MPC drive was able to function in real time for prediction horizons up to  $N = 8$ . The performance was maintained throughout transient occurrences due to target preconditioning offered by the projection algorithm. Partial to the success of achieving such a long prediction horizon was the selection of an optimal sampling frequency.

Performance of the IM drive, emulated on the real-time simulator, demonstrated the benefits of a long-horizon predictive controller, as the stator current and electromagnetic torque total demand distortions of 1.87% and 2.32% were recorded during steady-state operation. The response of the controller during transients was fast and maintained the drive parameters within or close to the rated values. In the case of the torque step, a settling time of 40ms was relatively slow due to the reaction of the PI controller. In the case of the speed reference step, a large speed error commanded a rapid torque adjustment, which the predictive controller reacted upon, and it delivered a maximum torque within 10ms.

Incorporating the MVA as solver to the FCS-MPC of multilevel inverters proved to be well suited for a high number of voltage levels. The real-time results showed that the MVA required a maximum of  $39\mu\text{s}$  out of the available  $83\mu\text{s}$  to solve the CVP associated with FCS-MPC of an eleven-level inverter. In steady-state conditions, such an inverter configuration almost matched the stator current and electromagnetic torque performances of a three-level inverter with a prediction horizon of eight. The higher number of voltage levels and switching devices respectively effected a reduction in terms of the inverter current TDD (6.91%) and the average device switching frequency (62Hz).



## 7.2 Publications

Based on the research work carried out during the completion of this study, the following publications resulted.

- J. Raath, H. du Toit Mouton and T. Geyer, “Integration of inverter constraints in geometrical quantification of the optimal solution to an MPC controller”, in *2016 IEEE 17<sup>th</sup> Workshop on Control and Modeling for Power Electronics (COMPEL)*, pp. 1-6, IEEE, 2016.
- J. Raath, H. du Toit Mouton, and Tobias Geyer, “On the Micciancio-Voulgaris algorithm to solve the long-horizon direct mpc optimization problem”, in *2017 IEEE International Symposium on Predictive Control of Electrical Drives and Power Electronics (PRECEDE)*, pp. 95-100, IEEE, 2017.
- J. Raath, H. du Toit Mouton, and Tobias Geyer. “Alternative Sphere Decoding algorithm for Long-horizon Model Predictive Control of Multi-level Inverters”, in *2020 IEEE 21<sup>st</sup> Workshop on Control and Modeling for Power Electronics (COMPEL)*, pp. 1-8, IEEE, 2020.

## 7.3 Future work

The outcomes of this research have demonstrated the potential of implementing long-horizon FCS-MPC in real time by using the proposed algorithms in a complex electrical IM drive. Based on the challenges ahead in deploying a commercially acceptable *long-horizon* MPC industrial drive, future research can be extended in the following directions:

1. Investigation into the flexibility of the SBDA in terms of matrix block size in conjunction with evolving processing technology. This can assist in extending the prediction horizon even further.
2. MPC of power electronic applications which pose a large lattice structure in relatively low dimensions can exploit the deterministic nature of the MVA, especially where processing technology with multiple cores is available.
3. Target preconditioning of the CVP is essential to mitigate the computational, worst-case-scenario, which usually occurs during transients. The projection algorithm

presented here as a preamble to solving the CVP will be a good match for implementation on an FPGA platform. This is supported by its fixed computational upper bound.

4. By reducing the online computations required in the implementation of long-horizon MPC, the orthogonality defect of the generator matrix as a general measure of the computational complexity and tuning parameter may be elaborated on.

# Appendices

# Appendix A

## Linear algebra

### A.1 Projection onto a vector

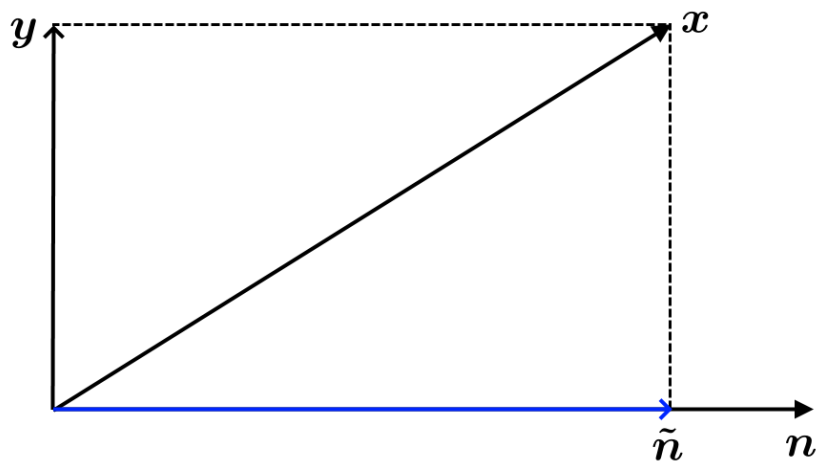


Figure A.1: Projection onto a vector in  $\mathbb{R}^2$  space.

In Euclidean geometry, the dot product or scalar product is the product of the Euclidean magnitudes of two non-zero vectors and the cosine of the angle between them. Of particular interest is the characteristic that when vectors are orthogonal to one another, the resulting dot product equals zero

$$\langle \mathbf{y} \cdot \mathbf{n} \rangle = \|\mathbf{y}\| \|\mathbf{n}\| \cos(90^\circ) = 0. \quad (\text{A.1a})$$

Consider vectors  $\mathbf{y}, \mathbf{n} \in \mathbb{R}^d$  with  $\mathbf{n} \neq 0$ . In the case where vector  $\mathbf{y}$  is orthogonal to  $\mathbf{n}$  there exists a unique decomposition whereby  $\mathbf{x} = \mathbf{y} + \tilde{\mathbf{n}}$ . If  $\tilde{\mathbf{n}} = k\mathbf{n}$  for some  $k \in \mathbb{R}$  then

$$0 = \langle \mathbf{y} \cdot \mathbf{n} \rangle \quad (\text{A.1b})$$

$$= \langle (\mathbf{x} - k\mathbf{n}) \cdot \mathbf{n} \rangle \quad (\text{A.1c})$$

$$= \langle \mathbf{x} \cdot \mathbf{n} \rangle - \langle k\mathbf{n} \cdot \mathbf{n} \rangle \quad (\text{A.1d})$$

$$k = \frac{\langle \mathbf{x} \cdot \mathbf{n} \rangle}{\langle \mathbf{n} \cdot \mathbf{n} \rangle} \quad (\text{A.1e})$$

which gives the orthogonal projection of  $\mathbf{x}$  onto  $\mathbf{n}$  as

$$\tilde{\mathbf{n}} = k\mathbf{n} = \frac{\langle \mathbf{x} \cdot \mathbf{n} \rangle}{\|\mathbf{n}\|^2} \mathbf{n}. \quad (\text{A.1f})$$

Interpreted simply, as the scaling of vector  $\mathbf{n}$  with a projection coefficient  $k$ .

## A.2 Hyperplanes

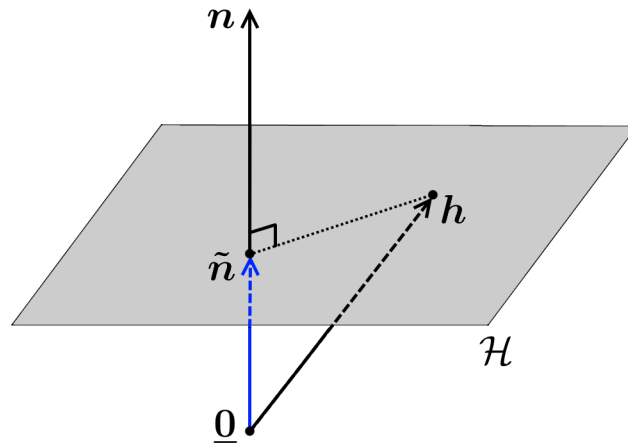


Figure A.2: A hyperplane in  $\mathbb{R}^3$  space.

A hyperplane is a subspace of dimension  $\mathbb{R}^{d-1}$  when residing in an ambient space of dimension  $\mathbb{R}^d$ . It can be described as a set, defined by a single scalar product equality in the form

$$\mathcal{H} = \{\mathbf{y} \in \mathbb{R}^d : \langle \mathbf{y} \cdot \mathbf{n} \rangle = p\}. \quad (\text{A.2a})$$

The normal vector to the hyperplane is denoted by  $\mathbf{n} \in \mathbb{R}^d, \mathbf{n} \neq \mathbf{0}$  with the translation of the hyperplane in the direction of  $\mathbf{n}$  denoted by  $p \in \mathbb{R}$ . When  $p = 0$  the hyperplane is centered around the origin  $\mathbf{0}$  and it is characterized by the set of vectors orthogonal to  $\mathbf{n}$ . This subspace is also referred to as the *null-space* of  $\mathbf{n}$ . Henceforth if a vector  $\mathbf{h}$  is known to reside in  $\mathcal{H}$ , the hyperplane can also be defined by

$$\mathcal{H} = \{\mathbf{y} \in \mathbb{R}^d : \langle (\mathbf{y} - \mathbf{h}) \cdot \mathbf{n} \rangle = 0, \mathbf{h} \in \mathcal{H}\} \quad (\text{A.2b})$$

which is the well known *Point normal form*. Normalising the normal vector with

$$\hat{\mathbf{n}} = \frac{\mathbf{n}}{\|\mathbf{n}\|} \quad (\text{A.2c})$$

allows for defining the hyperplane in *Hessian normal form*

$$\mathcal{H} = \{\mathbf{y} \in \mathbb{R}^d : \langle \mathbf{y} \cdot \hat{\mathbf{n}} \rangle = p \text{ with } \|\hat{\mathbf{n}}\| = 1\}. \quad (\text{A.2d})$$

In this form  $p$  uniquely gives the distance of the hyperplane from the origin, i.e.

$$p = \|\tilde{\mathbf{n}}\|. \quad (\text{A.2e})$$

### A.3 Projection onto a plane

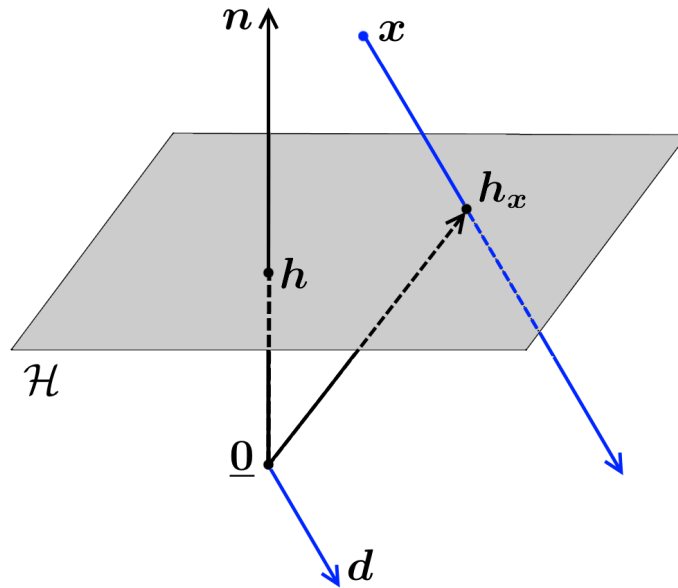


Figure A.3: Projection of a vector onto a plane in  $\mathbb{R}^3$  space.

Projection of a point or target  $\mathbf{x}$  onto a hyperplane can be accomplished through finding the point of intersection between the plane and the line describing the direction of projection. Consider a hyperplane  $\mathcal{H}$  defined in Point-normal form

$$\mathcal{H} = \{ \mathbf{y} \in \mathbb{R}^d : \langle (\mathbf{y} - \mathbf{h}) \cdot \mathbf{n} \rangle = 0, \mathbf{h} \in \mathcal{H} \}, \quad (\text{A.3a})$$

and a line  $\mathcal{L}$ , described as the set of points in a given direction from a point of origin, i.e.

$$\mathcal{L} = \{ \mathbf{y} \in \mathbb{R} : \mathbf{l}(y) = y\mathbf{d} + \mathbf{x}, \mathbf{x} \in \mathcal{L} \}. \quad (\text{A.3b})$$

The origin of the line is denoted by  $\mathbf{x}$ , the direction of the line by the vector  $\mathbf{d}$  and all scalars by  $y \geq 0$ . At the point of intersection  $\mathbf{y} = \mathbf{l}(y)$ . Substituting this into A.3a gives

$$0 = \langle (\mathbf{l}(y) - \mathbf{h}) \cdot \mathbf{n} \rangle \quad (\text{A.3c})$$

$$y = \frac{\langle (\mathbf{h} - \mathbf{x}) \cdot \mathbf{n} \rangle}{\langle \mathbf{d} \cdot \mathbf{n} \rangle}. \quad (\text{A.3d})$$

If  $\langle \mathbf{d} \cdot \mathbf{n} \rangle \neq 0$ , can  $y$  be calculated, suggesting that a single point of intersection exist.

Reinserting  $y$  into A.3b the point of intersection is then given by

$$\mathbf{h}_x = \frac{\langle (\mathbf{h} - \mathbf{x}) \cdot \mathbf{n} \rangle}{\langle \mathbf{d} \cdot \mathbf{n} \rangle} \mathbf{d} + \mathbf{x}. \quad (\text{A.3e})$$

## A.4 Transforming normals

A vector  $\mathbf{e}$ , orthogonal to any vector  $\mathbf{p}$  in a hyperplane  $P$  will result in

$$0 = \langle \mathbf{e} \cdot \mathbf{p} \rangle = \mathbf{e}^T \mathbf{p}, \quad \forall \mathbf{p} \in P. \quad (\text{A.4a})$$

Transforming  $\mathbf{p}$  by some transformation matrix  $\mathbf{H}$  and  $\mathbf{e}$  with another matrix  $\mathbf{H}'$ , while maintaining orthogonality, gives

$$0 = \langle \mathbf{H}' \mathbf{e} \cdot \mathbf{H} \mathbf{p} \rangle \quad (\text{A.4b})$$

$$= (\mathbf{H}' \mathbf{e})^T (\mathbf{H} \mathbf{p}) \quad (\text{A.4c})$$

$$= \mathbf{e}^T \mathbf{H}'^T \mathbf{H} \mathbf{p}. \quad (\text{A.4d})$$

In order for the expansion to hold, must  $\mathbf{H}'^T \mathbf{H} = \mathbf{I}$  and therefore

$$\mathbf{H}' = \mathbf{H}^{-T}, \quad (\text{A.4e})$$

which is known as the *standard transformation matrix for normals*.



## A.5 Hypercube elements

A  $d$ -dimensional hypercube can be interpreted as a convex hull defined by all sign permutations of the coordinates  $(\pm 1_1, \pm 1_2, \dots, \pm 1_d)$ . Any dimensional hypercube can be constructed from a single point through the process of adding dimensions and sweeping out volumes. Starting with a single point representing a hypercube of dimension zero, also referred to as a 0-cube, and moving this point to  $\pm 1$ , will sweep it into line segment or hypercube of dimension one (1-cube). If the line segment is moved with  $\pm 1$  in a perpendicular direction to itself, it sweeps out to a 2-dimensional square (2-cube). Similarly, a 3-cube is formed when the 2-cube is moved with  $\pm 1$  in a perpendicular direction to the plane it resides in. Congruently, this process can be generalized to any number of dimension. Fig. A.4 illustrate the process of sweeping out volumes from dimension 0 – 3. This process highlights the fact that every  $d$ -dimensional parent cube consist of a variety

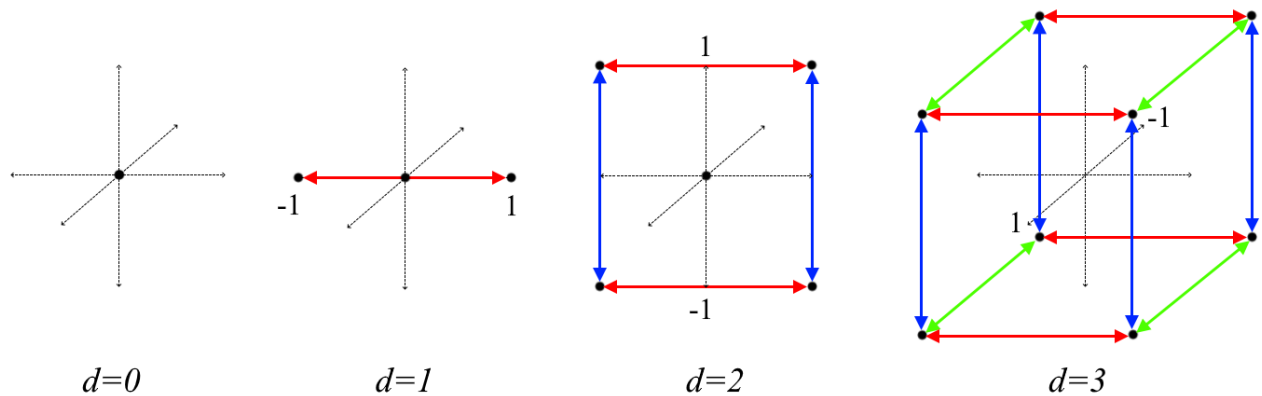


Figure A.4: Hypercubes in dimensions 0 to 3.

of child cubes in lower dimensions  $m$ . The most obvious is the  $2^d$ , zero-dimensional vertices and the  $2d$ ,  $(d - 1)$ -dimensional sides or faces [128]. All other  $m$ -dimensional cubes located on the boundary of a  $d$ -dimensional parent cube, can be quantified by the identity [73]

$$x = 2^n \frac{d!}{m!n!} \quad \text{with} \quad n = d - m. \quad (\text{A.5})$$

Note that  $n!$  denotes the factorial of  $n$ . Table A.5 list various  $d$ -dimensional cubes with their child-cubes, demonstrating the arising complexity of such a simple shape into higher dimensions.

<b>d</b>			0	1	2	3	4	5	6	7	8	9	10
	<b>d-cube</b>	<b>Names</b>	<b>Vertex 0-face</b>	<b>Edge 1-face</b>	<b>Plane 2-face</b>	<b>Cell 3-face</b>	<b>4-face</b>	<b>5-face</b>	<b>6-face</b>	<b>7-face</b>	<b>8-face</b>	<b>9-face</b>	<b>10-face</b>
0	0-cube	Point	1										
1	1-cube	Line	2	1									
2	2-cube	Square	4	4	1								
3	3-cube	Cube	8	12	6	1							
4	4-cube	Tesseract	16	32	24	8	1						
5	5-cube	Penteract	32	80	80	40	10	1					
6	6-cube	Hexeract	64	192	240	160	60	12	1				
7	7-cube	Hepteract	128	448	672	560	280	84	14	1			
8	8-cube	Octeract	256	1024	1792	1792	1120	448	112	16	1		
9	9-cube	Enneract	512	2304	4608	5376	4032	2016	672	144	18	1	
10	10-cube	Dekeract	1024	5120	11520	15360	13440	8064	3360	960	180	20	1

Figure A.5: Hypercube elements

## A.6 Dual interpretation of projection

The minimum distance from a point  $\mathbf{x} \in \mathbb{R}^d$  to a convex set  $\mathcal{P} \in \mathbb{R}^d$  can be determined by maximising the distance from  $\mathbf{x}$  to a hyperplane  $\partial\mathcal{H}$  from the set of hyperplanes which separates  $\mathbf{x}$  from the convex set  $\mathcal{P}$ . If a separating hyperplane exists, then it is assumed that  $\mathbf{x}$  is exterior to the set  $\mathcal{P}$  or exists on the set's boundary. [129].

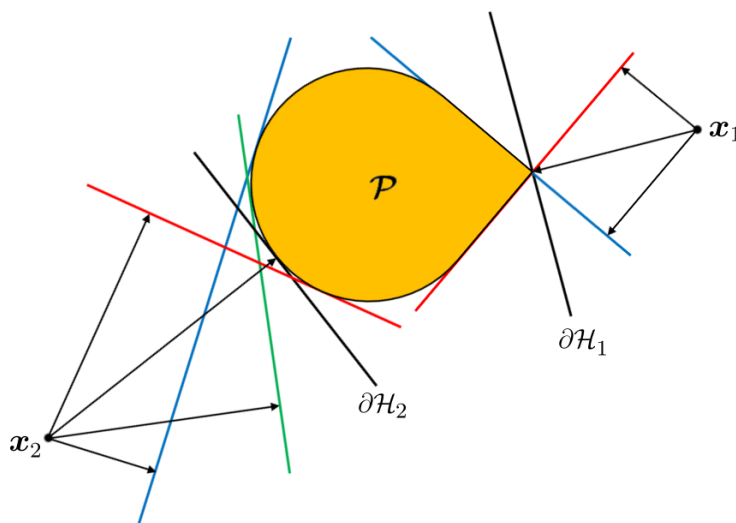


Figure A.6: Dual interpretation of projection of points  $\mathbf{x}_{1,2}$  onto the convex set  $\mathcal{P} \in \mathbb{R}^2$ .

## A.7 Orthogonal decomposition

Let  $S$  be a subspace of  $\mathbb{R}^d$ . If the vector  $\mathbf{x}_S \in S$  is closest to vector  $\mathbf{x} \in \mathbb{R}^d$  then the difference  $\mathbf{x} - \mathbf{x}_S$  is in  $S^\perp$  and orthogonal to the vectors in  $S$ . Thus if  $\mathbf{x}_{S^\perp} = \mathbf{x} - \mathbf{x}_S$  then  $\mathbf{x} = \mathbf{x}_S + \mathbf{x}_{S^\perp}$  where  $\mathbf{x}_S \in S$  and  $\mathbf{x}_{S^\perp} \in S^\perp$ . [130, p.134]

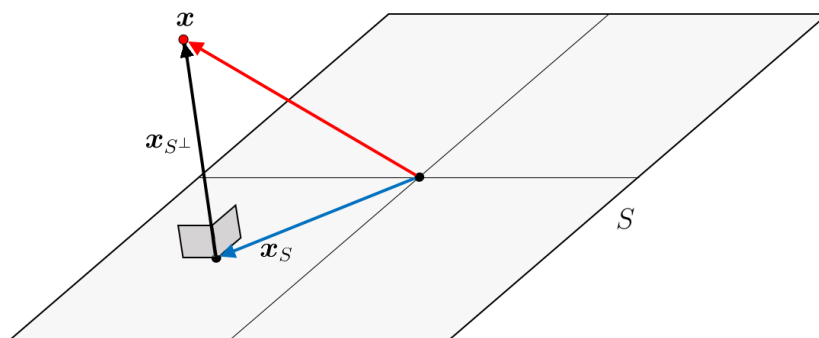


Figure A.7: Orthogonal decomposition of vector  $\mathbf{s} \in \mathbb{R}^3$  with subspace  $S \in \mathbb{R}^2$ .

# Appendix B

## Algorithms and processes

### B.1 Adapted projection algorithm

The following pseudo code is adapted from Algorithm 4 to facilitate projection onto the lattice convex hull of multilevel inverters ( $l_v \geq 3$ ).

---

**Algorithm 8** *Projection algorithm*


---

```

1: function PROJH( $\mathbf{x}, \mathbf{H}, \mathbf{N}, l_v$ )
2:    $\tilde{\mathbf{x}} \leftarrow \mathbf{x}$ 
3:    $\hat{\mathbf{p}} \leftarrow \mathbf{0}$ 
4:    $\hat{k} \leftarrow \infty$ 
5:    $b \leftarrow (l_v - 1)/2$ 
6:   while  $|\hat{k}| > b$  do
7:      $\mathbf{k} = \mathbf{N}^T \mathbf{x}$ 
8:      $j^* = \arg \max_j ( (|k_j| - b) / \|\mathbf{n}_j\| )$  for all  $j = 1, 2, \dots, d$ 
9:      $\hat{k} = k_{j^*}$ 
10:     $s = \text{sign}(\hat{k})$ 
11:     $\hat{\mathbf{n}} = s \mathbf{n}_{j^*}$ 
12:     $\hat{\mathbf{h}} = s \mathbf{h}_{j^*}$ 
13:    if  $|\hat{k}| > b$  then
14:       $\mathbf{p}^\perp = -((|\hat{k}| - b) / \langle \hat{\mathbf{n}} \cdot \hat{\mathbf{n}} \rangle) \hat{\mathbf{n}}$ 
15:       $\mathbf{x} = \mathbf{x} + \mathbf{p}^\perp - \hat{\mathbf{h}}$ 
16:       $\mathbf{H}_{(:,j^*)} = \mathbf{0}$ 
17:       $\mathbf{N} = \mathbf{N} - \hat{\mathbf{n}} ((\mathbf{N}^T \hat{\mathbf{n}}) / \langle \hat{\mathbf{n}} \cdot \hat{\mathbf{n}} \rangle)^T$ 
18:       $\hat{\mathbf{p}} = \hat{\mathbf{p}} + \mathbf{p}^\perp$ 
19:    end if
20:  end while
21:  return  $\tilde{\mathbf{x}} = \tilde{\mathbf{x}} + \hat{\mathbf{p}}$ 
22: end function

```

---

## B.2 Acquiring the output reference variables

This section is a capitulation of the work published in [5], altered to suit the variables in this research.

The dynamic model of the plant (6.7) is converted to the synchronously rotating  $dq$ -frame, with  $\omega_{fr} = \omega_s$ . In the plant model of (6.7), the first two differential equations are based on the  $LC$ -filter equations, which when transformed to the rotating  $dq$ -frame, result in

$$\frac{d\mathbf{i}_i}{dt} = \left( \frac{-R_c - R_l}{L} - \omega_s \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \right) \mathbf{i}_i - \frac{1}{L} \mathbf{v}_c + \frac{R_c}{L} \mathbf{i}_s + \frac{1}{L} \mathbf{v}_i \quad (\text{B.1a})$$

$$\frac{d\mathbf{v}_c}{dt} = \frac{1}{C} \mathbf{i}_i - \omega_s \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \mathbf{v}_c - \frac{1}{C} \mathbf{i}_s. \quad (\text{B.1b})$$

The remaining differential equations of (6.7) are defined by the IM model equations (2.65) which are already formulated in the  $dq$ -frame. Augmented with (6.5) and setting  $\mathbf{v}_r = 0$  gives

$$\frac{d\mathbf{i}_s}{dt} = \frac{R_c L_r}{D} \mathbf{i}_i + \frac{L_r}{D} \mathbf{v}_c - \left( \frac{1}{\tau_s} + \frac{R_c L_r}{D} + \omega_s \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \right) \mathbf{i}_s + \left( \frac{1}{\tau_r} - \omega_r \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \right) \frac{L_m}{D} \boldsymbol{\psi}_r \quad (\text{B.1c})$$

$$\frac{d\boldsymbol{\psi}_r}{dt} = \frac{L_m}{\tau_r} \mathbf{i}_s - \left( \frac{1}{\tau_r} + (\omega_s - \omega_r) \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \right) \boldsymbol{\psi}_r. \quad (\text{B.1d})$$

As a result, the state of the plant in continuous-time state-space is given by

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{F}_{dq} \mathbf{x}(t) + \mathbf{G}_{dq} \mathbf{v}_i(t) \quad (\text{B.2})$$

with the respective state and input matrices

$$\mathbf{F}_{dq} = \begin{bmatrix} \frac{-R_c - R_l}{L} \mathbf{I}_2 - \omega_s \mathbf{J} & -\frac{1}{L} \mathbf{I}_2 & \frac{R_c}{L} \mathbf{I}_2 & \mathbf{0}_2 \\ \frac{1}{C} \mathbf{I}_2 & -\omega_s \mathbf{J} & -\frac{1}{C} & \mathbf{0}_2 \\ \frac{R_c L_r}{D} \mathbf{I}_2 & \frac{L_r}{D} \mathbf{I}_2 & -\left( \frac{1}{\tau_s} + \frac{R_c L_r}{D} + \omega_s \mathbf{J} \right) \mathbf{I}_2 & \left( \frac{1}{\tau_r} \mathbf{I}_2 - \omega_r \mathbf{J} \right) \frac{L_m}{D} \\ \mathbf{0}_2 & \mathbf{0}_2 & \frac{L_m}{\tau_r} \mathbf{I}_2 & -\frac{1}{\tau_r} \mathbf{I}_2 - (\omega_s - \omega_r) \mathbf{J} \end{bmatrix}, \quad (\text{B.3})$$

$$\mathbf{G}_{dq} = \begin{bmatrix} \frac{1}{L} \mathbf{I}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \end{bmatrix}^T \mathbf{K}. \quad (\text{B.4})$$

During *steady-state* conditions the derivative of (B.2) is equal to zero, which allows for expressing the state-vector in terms of the inverter voltage

$$\mathbf{x}(t) = -\mathbf{F}_{dq}^{-1} \mathbf{G}_{dq} \mathbf{v}_i(t). \quad (\text{B.5})$$

To compute the inverter voltage  $\mathbf{v}_i$  we select the stator flux as output to the plant

$$\boldsymbol{\psi}_s(t) = \mathbf{C}_{dq} \mathbf{x}(t). \quad (\text{B.6})$$

Calling upon (2.62a), which portrays the relationship between the stator flux and state-variables, i.e. stator current and rotor flux, the output matrix is defined

$$\mathbf{C}_{dq} = \begin{bmatrix} \mathbf{0}_2 & \mathbf{0}_2 & \frac{D}{L_r} \mathbf{I}_2 & \frac{L_m}{L_r} \mathbf{I}_2 \end{bmatrix}. \quad (\text{B.7})$$

Substituting (B.5) and (B.7) into (B.6) and with some rearranging leaves

$$\mathbf{v}_i(t) = -(\mathbf{C}_{dq} \mathbf{F}_{dq}^{-1} \mathbf{G}_{dq})^{-1} \boldsymbol{\psi}_s(t) \quad (\text{B.8})$$

and as a result, the state-vector in terms of the stator flux

$$\mathbf{x}(t) = \mathbf{F}_{dq}^{-1} \mathbf{G}_{dq} (\mathbf{C}_{dq} \mathbf{F}_{dq}^{-1} \mathbf{G}_{dq})^{-1} \boldsymbol{\psi}_s(t). \quad (\text{B.9})$$

Computing  $\mathbf{x}(t)$  for steady-state conditions, i.e. constant rotor speed  $\omega_r$ , requires the stator flux  $\boldsymbol{\psi}_s$  and angular speed  $\omega_s$ . With the rotor flux vector aligned with the  $d$ -axis of the reference frame, it allows for the reformulation thereof in terms of its reference magnitude  $\Psi_r$ , i.e.

$$\boldsymbol{\psi}_r = [\Psi_r \ 0]^T. \quad (\text{B.10})$$

Substitution of (B.10) into the torque equation (2.64) gives the  $q$ -component of the stator flux

$$\psi_{sq} = \text{pf} \frac{T_e}{\Psi_r} \frac{D}{X_m}. \quad (\text{B.11})$$

Maintaining steady state conditions, effects the derivative of (2.63b) to be zero

$$0 = R_r \frac{X_m}{D} \boldsymbol{\psi}_s - R_r \frac{X_s}{D} \boldsymbol{\psi}_r - (\omega_s - \omega_r) \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \boldsymbol{\psi}_r + \mathbf{v}_r. \quad (\text{B.12})$$

Dropping  $\mathbf{v}_r$  and substituting (B.10) into the  $d$ -part of (B.12), result in the corresponding  $d$ -component of the stator flux

$$\psi_{sd} = \frac{X_s}{X_m} \Psi_r. \quad (\text{B.13})$$

Substitution of (B.10) and (B.13) into the  $q$ -part of (B.12) gives the angular stator speed

$$\omega_s = \omega_r + \text{pf} \frac{R_r T_e}{\Psi_r^2}. \quad (\text{B.14})$$

Henceforth, with all the variables computable, the state-vector (B.9), can be calculated for a rotor-flux setting  $\Psi_r$  with torque command  $T_e$  at an angular rotor speed  $\omega_r$ .

From the computed state-vector  $\mathbf{x}(t)$ , extracting of the output reference vector follows

$$\mathbf{y}^*(t) = \mathbf{C}_{dq} \mathbf{x}(t) \quad \text{with} \quad \mathbf{C}_{dq} = [\mathbf{I}_6 \ \mathbf{0}_{6 \times 2}]. \quad (\text{B.15})$$

Importantly  $\mathbf{y}^*(t)$  is conceptualized in the  $dq$ -frame and subsequently require conversion to the stationary  $\alpha\beta$ -frame for every step in the prediction horizon. Note that each pair of  $dq$  coordinates in the output vector  $\mathbf{y}^*$  must *individually* be converted with the rotation matrix  $\mathbf{R}$  (2.40) before the sequence of reference output vectors can be put together

$$\tilde{\mathbf{y}}^*(t) = \left[ \mathbf{y}^{*T}(t + 1T_s) \quad \mathbf{y}^{*T}(t + 2T_s) \dots \mathbf{y}^{*T}(t + NT_s) \right]^T. \quad (\text{B.16})$$

# Appendix C

## Real-time simulation model

The real-time simulation system consists out of three subsystems, namely, the console subsystem (SC\_console), the master subsystem (SM\_system) and the slave subsystem (SS\_system). From the console or user interface, the rotor flux reference and rotor speed reference, i.e. drive settings, are supplied to the master subsystem or controller. The mechanical load torque ( $T_m$ ), recording signal (rec) and load type selection (load\_sel) are passed through the controller to the slave subsystem or plant. Actuation of the plant is in the form of the inverter control vector (uabc), with measurements of the system's response, i.e. state returning to the input of the controller. The (gui)-output supplies the user interface with feedback such as the motor speed and torque exerted. Initiation of the system parameters, precomputation of the system matrices and storage thereof in look-up tables are facilitated by the load\_setup\_v6.m file called in the Model-initialisation block.



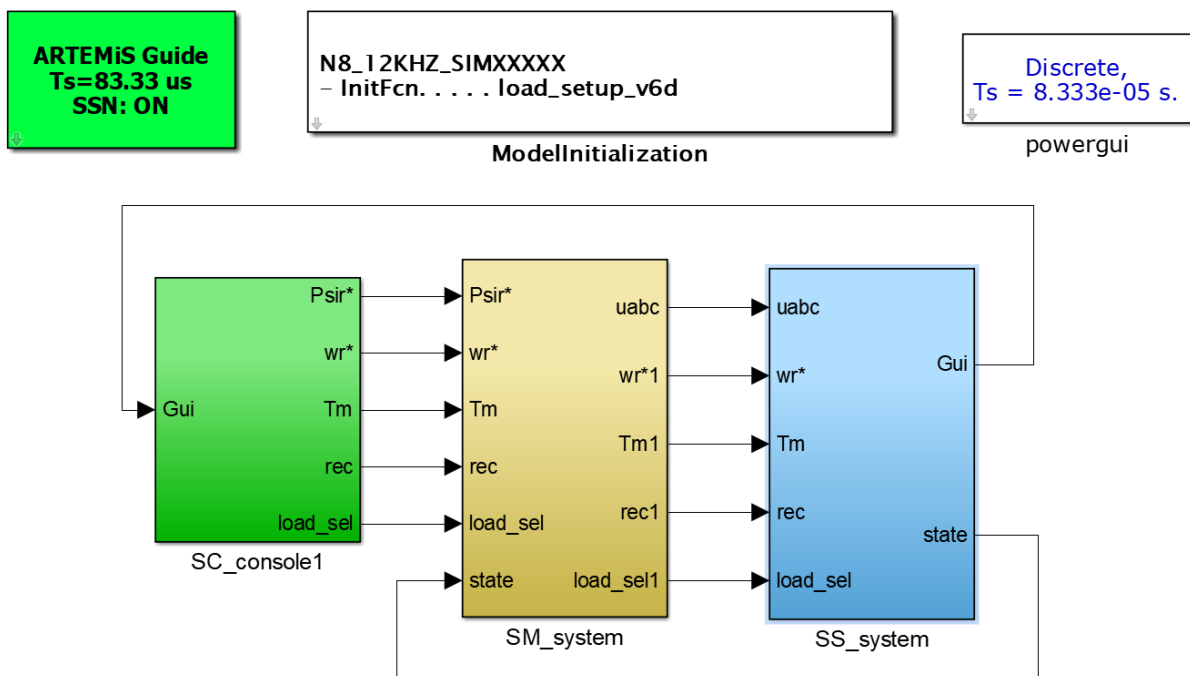


Figure C.1: Drive system model.

### C.0.1 Console sub-system

The user interface is available on the host computer while the simulation is in progress. This allows the user to adjust parameters such as the rotor speed reference ( $\omega_r^*$ ) and load torque ( $T_m$ ) if the “Mode select” is on manual. If set to “Auto” then the parameters are varied according to a preset sequence programmed in the (Seq1)-block. The load selector distinguish between the type of mechanical load which can be applied to the induction machine. The rotor flux reference ( $\Psi_r^*$ ) is kept at its nominal value for maximum excitation of the induction machine.

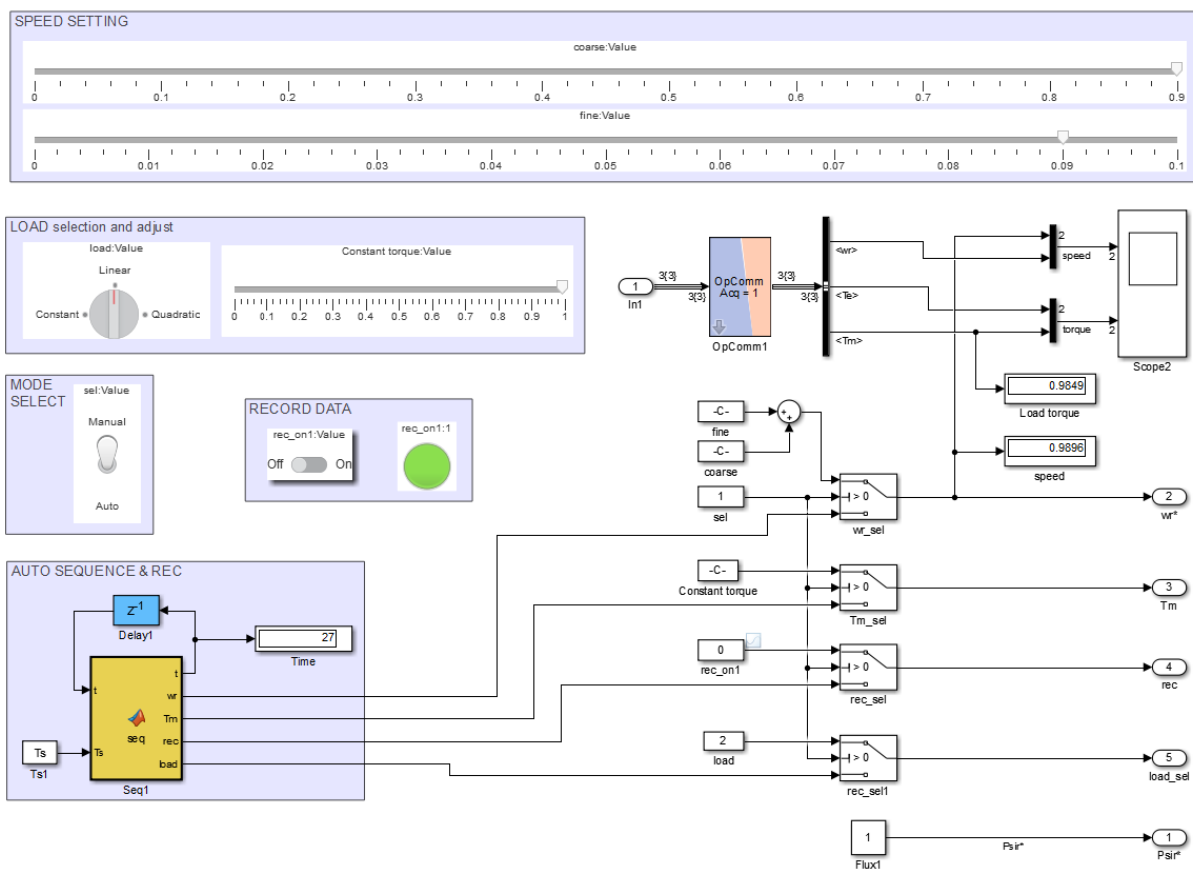


Figure C.2: Console sub-system: User interface.

## C.0.2 Master sub-system

The master sub-system contains a conversion subsystem, the PI speed controller and the FCS-MPC subsystem. In the conversion subsystem, the measured system states are converted to the stationary  $\alpha\beta$  frame. The values are also used to compute the estimated rotor flux and it's relative angular position. The PI speed controller determines the torque reference ( $T_e^*$ ).

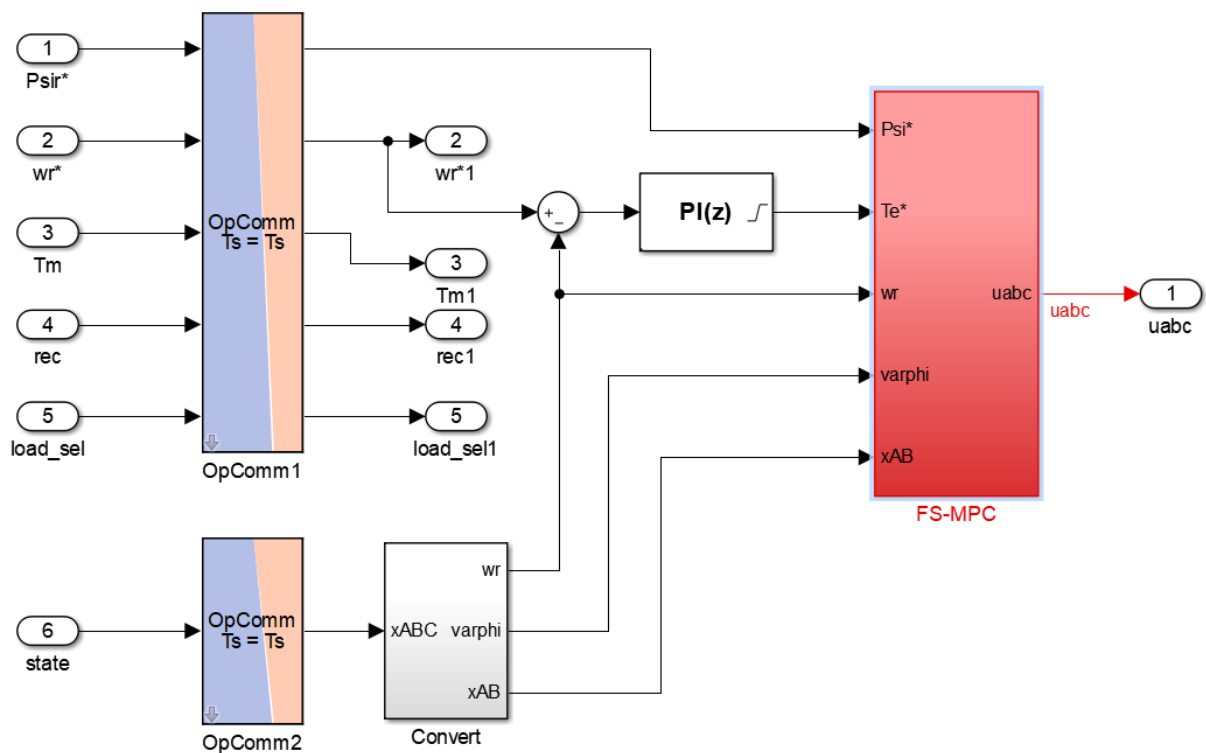


Figure C.3: Master sub-system: PI and predictive controller.

### C.0.2.1 Conversion and flux estimation

The measured system states, i.e. inverter current, capacitor voltage and stator current are converted to their equivalent  $\alpha\beta$  values. From the stator current and positional information of the tacho generator ( $\theta$ ), the rotor flux ( $\psi_{sr}$ ) and its relative angular position ( $\varphi$ ) is estimated. The multiplexer combines the system states and outputs the system state vector ( $X_{AB}$ ) to the FCS-MPC subsystem.

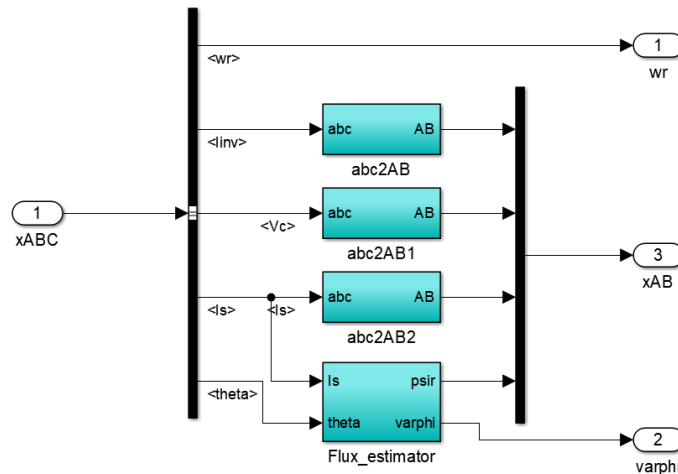


Figure C.4: Conversion and estimation of the system states.

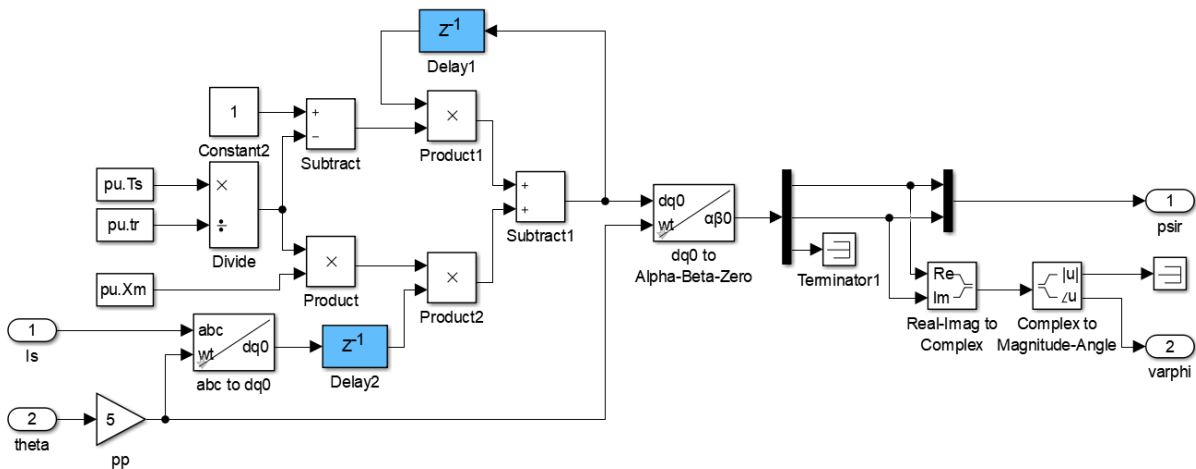


Figure C.5: Rotor flux estimation

### C.0.2.2 Predictive controller

The FCS-MPC process is sectioned into five subsystems, the first (LUTs), contains the precomputed look-up tables, the second (Ref\_gen), computes the output references, the third (XAB\_est), extrapolate the system states, the fourth (Uunc), finds the unconstrained minimum and the fifth (Optimize), solves the ILS problem. An optimal control vector ( $u_{abc}$ ), emerges as the output.

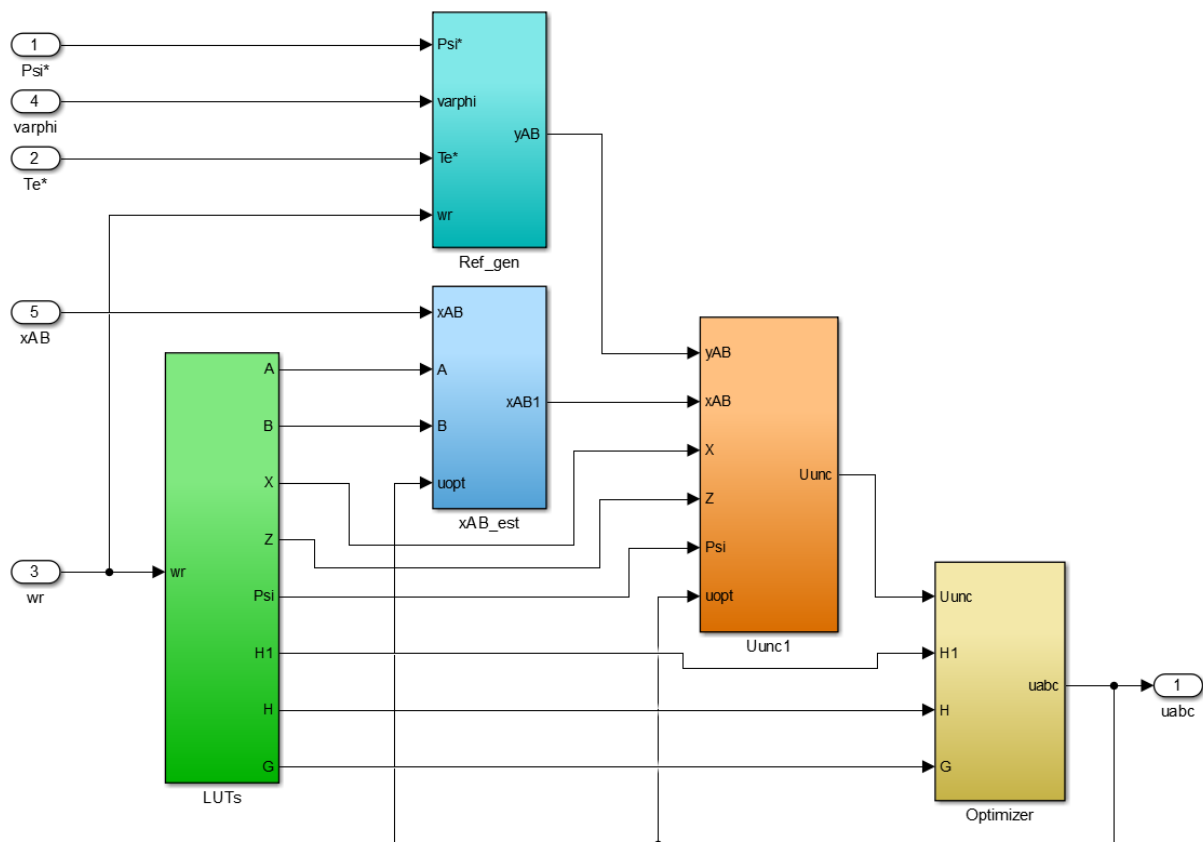


Figure C.6: FCS-MPC process.

The subsystem containing the look-up tables (LUTs) receive the rotor speed as input, discretize it to the nearest  $1/100^{th}$  of the rated speed and use it select the appropriate set of matrices. Figure C.7 shows the expanded subsystem. Computation of the time-varying output references ( $y_{AB}$ ), over the prediction horizon is accomplished in the (Ref-gen)-subsystem. Figure C.8 expands the subsystem which conceptualizes the output references in the rotating  $dq$ -frame and then converts the individual references to the stationary  $\alpha\beta$ -frame.

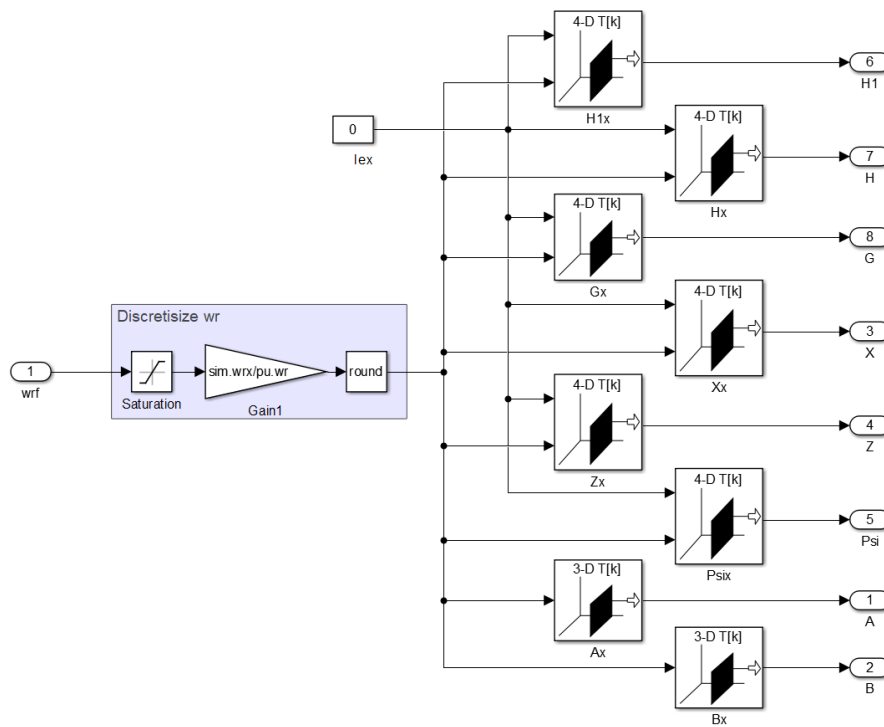


Figure C.7: Look-up tables selected from discretized speed measurement.

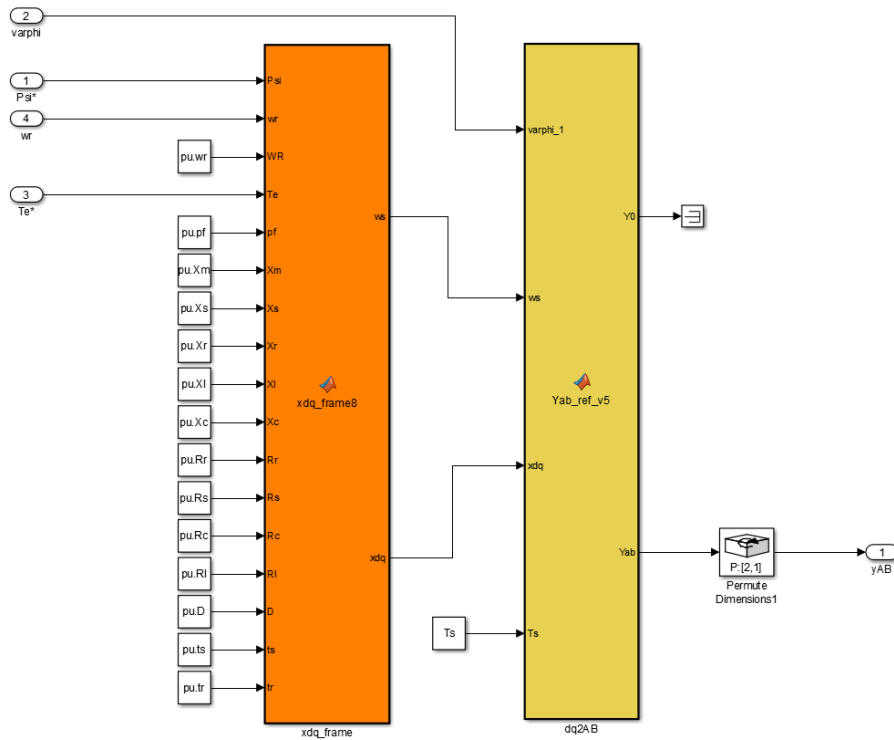


Figure C.8: Compute output references in  $dq$ -rotating frame and convert to stationary Alpha-Beta frame.

Subsystem ( $x_{AB\_est}$ ) expanded in Figure C.9, extrapolate the system states to compensate for the  $2T_s$  delay in the acquisition and actuation processes. Subsystem (Uunc), expanded in Figure C.10 computes the unconstrained minimum from the previous control action and the appropriate matrices supplied by the (LUT)-subsystem.

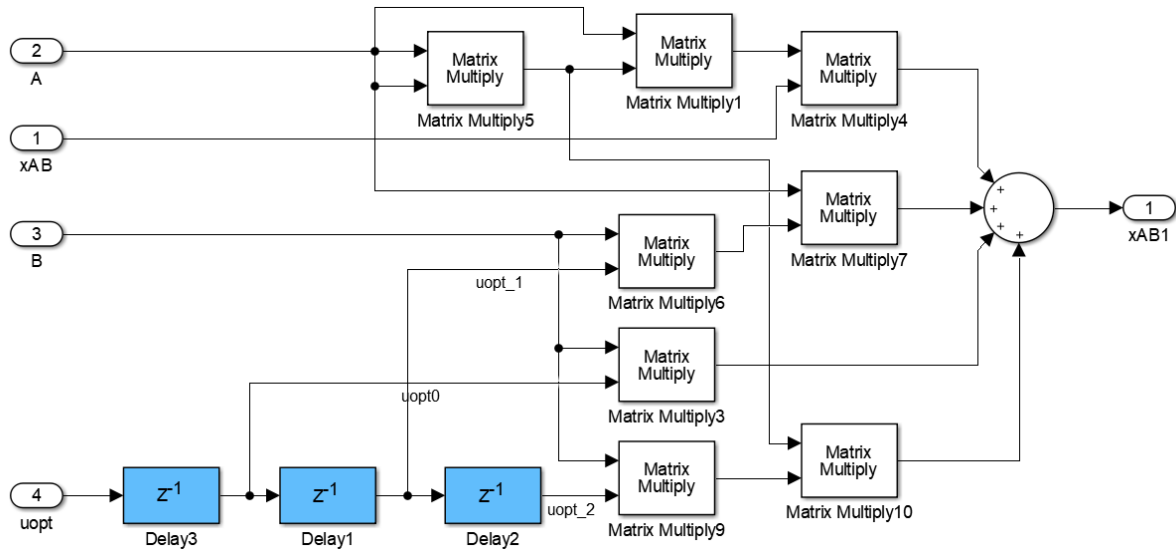


Figure C.9: Extrapolation of the system states.

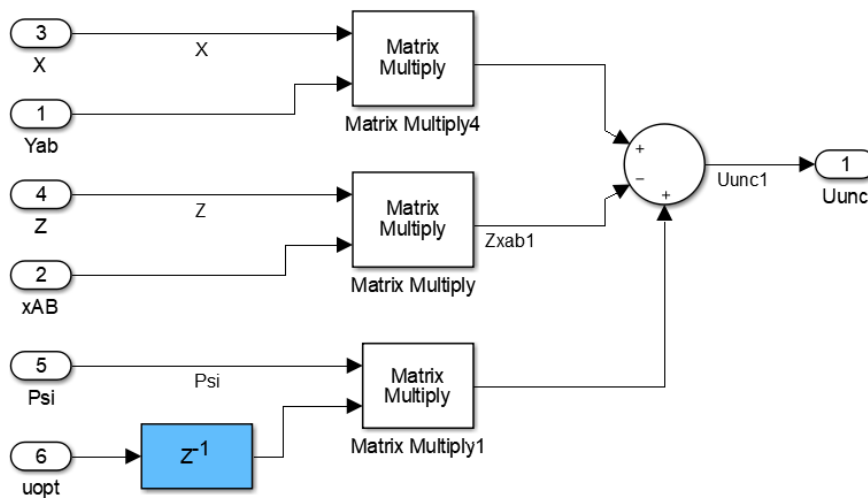


Figure C.10: Calculate the unconstrained minimum.

The (Optimizer)-subsystem optimizes the FCS-MPC problem. First, the unconstrained minimum is transformed to the  $\mathbf{H}$ -coordinate space where it is subjected to target preconditioning in subsystem (ProjH). In this subsystem the proposed projection algorithm (5) is invoked to give the updated target ( $\mathbf{x}^*$ ). Calculation of the initial sphere-radius occurs in the (BABAI)-sybsystem shown in Figure C.12. The (SBDA)-subsystem calls the proposed SBDA (Algorithm 7), which solves the closest vector problem, applies the receding-horizon strategy and outputs the optimal index vector (popt) that selects from the inverter control set (lookup table U), the optimum inverter control vector (uabc).

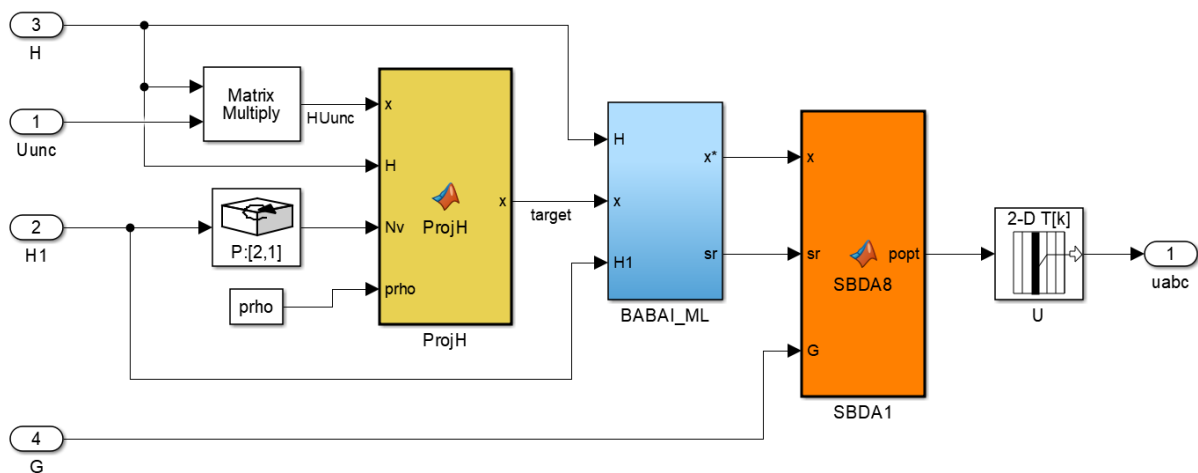


Figure C.11: Optimising the ILS problem.

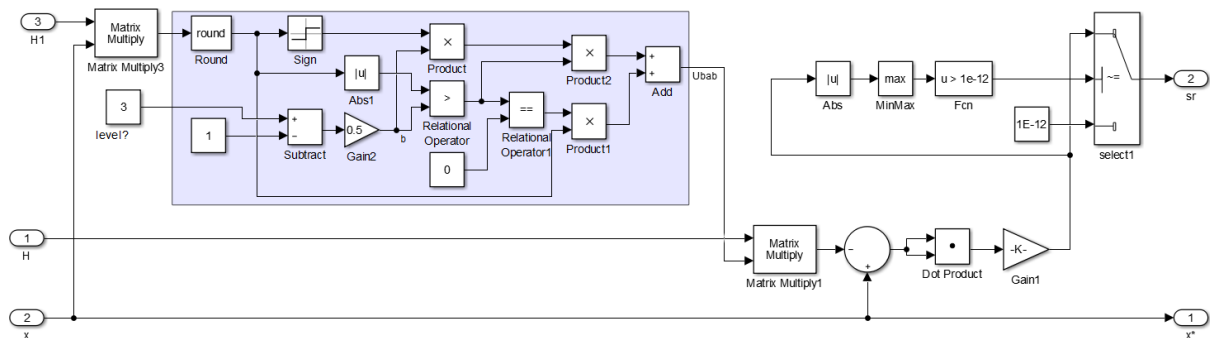


Figure C.12: Initial sphere radius computed from the Babai estimate.



### C.0.3 Slave sub-system

The slave sub-system consists of three subsystems, the plant (Plant), the input (ADC) and output (DAC) ports (IOs), and the data recording subsystem (Data\_capture). Before actuation of the plant take place, the control vector ( $u_{abc}$ ) passes through the IO-subsystem and the external hardwired loop. Upon actuation, the plant response also exits and returns through the IO-block before it is combined into the system (state)-bus. The (Gui)-bus conveys the mechanical load torque ( $T_m$ ), the electromechanical torque ( $T_e$ ) and rotational speed of the induction machine. The data capture block records the system inputs and outputs as commanded by the (rec)-signal from the user interface.

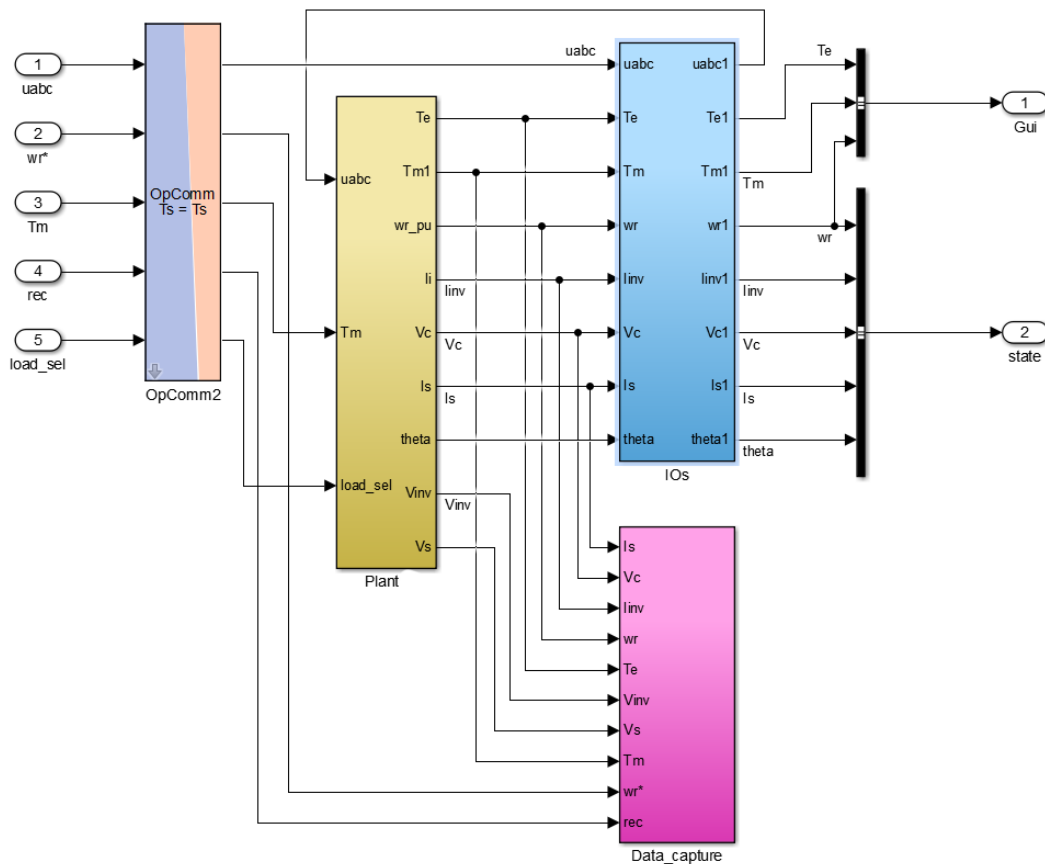


Figure C.13: Slave sub-system with Plant, I/Os and Data capture subsystems.

C.0.3.1 Plant

The (Plant)-subsystem contains the emulated IM drive with the NPC inverter, the *LC*-filter and squirrel-cage induction motor realised with OPAL-RT's state-space nodal (SSN) components. A gating subsystem, expanded in Figure C.15, inhibits illegal switching actions and converts the inverter control vector to an appropriate gating vector for the NPC. The (Variable\_load)-subsystem applied the mechanical torque ( $T_m$ ) as demanded from the user interface via the (load\_sel)-signal.

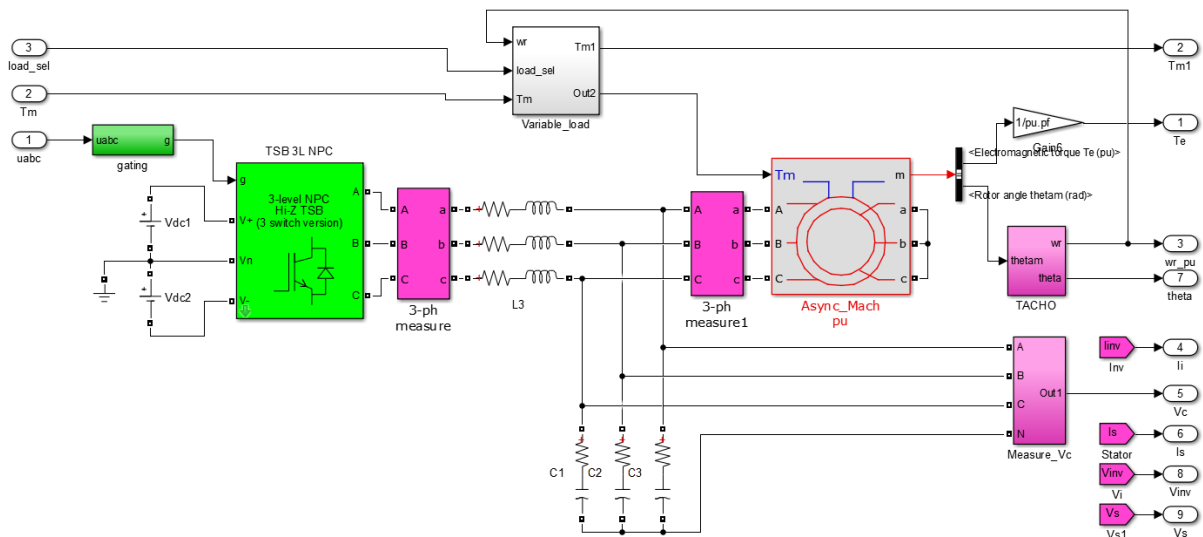


Figure C.14: Plant: NPC inverter driving the IM via an intermediate *LC* filter.

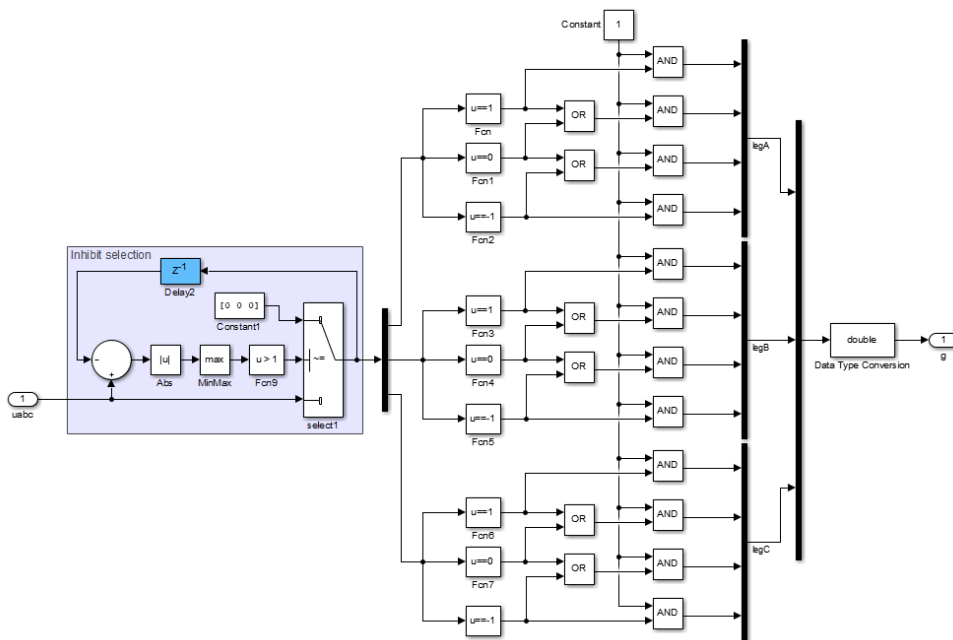


Figure C.15: NPC inverter gating control.

### C.0.3.2 Input/output external loop

The (IO)-subsystem contains the OPAL-RT function blocks that manage the DAC and ADC interfaces. Note that a 1pu signal translates to a 1V analogue signal. To improve the signal to noise ratio over the external loop, the pu-values were amplified with a factor ten before analogue transmission. After acquiring/sampling the analogue signals the quantified values were reduced with a factor ten to give the values in per unit. The (Calibrate)-subsystem was inserted with amplification factors for each individual ADC to improve the accuracy of the overall conversion process.

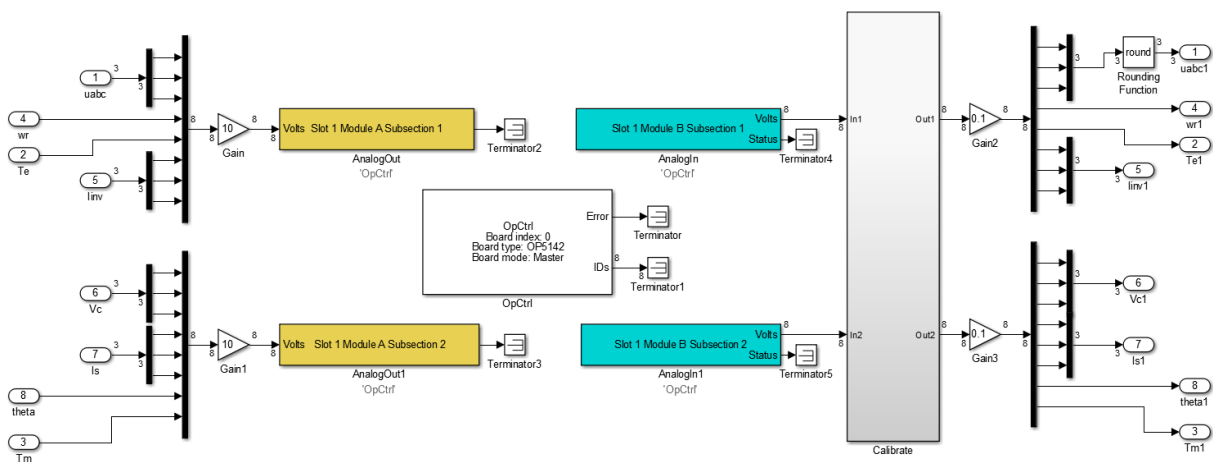


Figure C.16: Digital-to-analogue and analogue-to-digital conversion via the external hardware loop.

# Bibliography

- [1] S. Kouro, P. Cortés, R. Vargas, U. Ammann, and J. Rodríguez, “Model predictive control - A simple and powerful method to control power converters,” *Industrial Electronics, IEEE Transactions on*, vol. 56, no. 6, pp. 1826–1838, 2009.
- [2] P. Agrawal, S. Dubey, and S. Bharti, “Comparative study of fuzzy logic based speed control of multilevel inverter fed brushless dc motor drive,” *International Journal of Power Electronics and Drive Systems*, vol. 4, no. 1, p. 70, 2014.
- [3] E. Colak, I. Kabalci and B. Ramazan, “Review of multilevel voltage source inverter topologies and control schemes,” *Energy conversion and management*, vol. 52, no. 2, pp. 1114–1128, 2011.
- [4] J. Rodriguez, J. Lai, and F. Peng, “Multilevel inverters: a survey of topologies, controls, and applications,” *IEEE Transactions on industrial electronics*, vol. 49, no. 4, pp. 724–738, 2002.
- [5] T. Geyer, *Model predictive control of high power converters and industrial drives*. John Wiley & Sons, 2016.
- [6] J. Rodriguez and P. Cortes, *Predictive Control of Power Converters and Electrical Drives*. John Wiley & Sons, Ltd., 2012.
- [7] L. Westphal, *Handbook of Control Systems Engineering*. Boston, MA: Springer US, 2001.
- [8] D. Quevedo, R. Aguilera, and T. Geyer, *Predictive Control in Power Electronics and Drives: Basic Concepts, Theory, and Methods*, pp. 181–226. Springer International Publishing, 2014.
- [9] J. Mossoba and P. Lehn, “A controller architecture for high bandwidth active power filters,” *IEEE transactions on power electronics*, vol. 18, no. 1, pp. 317–325, 2003.
- [10] P. Mattavelli, “An improved deadbeat control for ups using disturbance observers,” *IEEE Transactions on Industrial Electronics*, vol. 52, no. 1, pp. 206–212, 2005.

- [11] A. Nasiri, "Digital control of three-phase series-parallel uninterruptible power supply systems," *IEEE transactions on power electronics*, vol. 22, no. 4, pp. 1116–1127, 2007.
- [12] P. Correa, M. Pacas, and J. Rodriguez, "Predictive torque control for inverter-fed induction machines," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 2, pp. 1073–1079, 2007.
- [13] I. Takahashi and T. Noguchi, "A new quick-response and high-efficiency control strategy of an induction motor," *IEEE Transactions on Industry applications*, no. 5, pp. 820–827, 1986.
- [14] T. Noguchi, H. Tomiki, S. Kondo, and I. Takahashi, "Direct power control of pwm converter without power-source voltage sensors," *IEEE Transactions on Industry Applications*, vol. 34, no. 3, pp. 473–479, 1998.
- [15] M. Depenbrock, "Direct self-control (dsc) of inverter-fed induction machine," *IEEE transactions on Power Electronics*, vol. 3, no. 4, pp. 420–429, 1988.
- [16] E. Flach, R. Hoffmann, and P. Mutschler, "Direct mean torque control of an induction motor," in *European Conference on Power Electronics and Applications*, vol. 3, pp. 3–672, Proceedings published by various publishers, 1997.
- [17] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, "Survey constrained model predictive control: Stability and optimality," *Automatica (Journal of IFAC)*, vol. 36, no. 6, pp. 789–814, 2000.
- [18] S. Vazquez, J. Leon, L. Franquelo, J. Rodriguez, H. Young, A. Marquez, and P. Zanchetta, "Model predictive control: A review of its applications in power electronics," *IEEE Industrial Electronics Magazine*, vol. 8, no. 1, pp. 16–31, 2014.
- [19] J. Holtz, "A predictive controller for the stator current vector of ac machines fed from a switched voltage source," *Proc. of IEE of Japan IPEC-Tokyo'83*, pp. 1665–1675, 1983.
- [20] T. Geyer, "A comparison of control and modulation schemes for medium-voltage drives: Emerging predictive control concepts versus pwm-based schemes," *IEEE Transactions on Industry Applications*, vol. 47, no. 3, pp. 1380–1389, 2011.
- [21] B. Zhang, X. Wang, Y. Tang, and W. Zhang, "Predictive functional control and stability analysis of mimo bilinear systems," in *2006 American Control Conference*, pp. 5–13, IEEE, 2006.

- [22] R. Aguilera and D. Quevedo, "On stability and performance of finite control set mpc for power converters," in *2011 Workshop on Predictive Control of Electrical Drives and Power Electronics*, pp. 55–62, IEEE, 2011.
- [23] R. Aguilera and D. Quevedo, "Stability analysis of quadratic mpc with a discrete input alphabet," *IEEE Transactions on Automatic Control*, vol. 58, no. 12, pp. 3190–3196, 2013.
- [24] T. Geyer, P. Karamanakos, and R. Kennel, "On the benefit of long-horizon direct model predictive control for drives with lc filters," in *2014 IEEE Energy Conversion Congress and Exposition (ECCE)*, pp. 3520–3527, IEEE, 2014.
- [25] T. Geyer and D. Quevedo, "Performance of multistep finite control set model predictive control for power electronics," *IEEE Transactions on Power Electronics*, vol. 30, no. 3, pp. 1633–1644, 2015.
- [26] J. Rodriguez, M. Kazmierkowski, J. Espinoza, P. Zanchetta, H. Abu-Rub, H. Young, and C. Rojas, "State of the art of finite control set model predictive control in power electronics," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 2, pp. 1003–1016, 2013.
- [27] T. Geyer, G. Papafotiou, and M. Morari, "Model predictive control in power electronics: A hybrid systems approach," in *Proceedings of the 44th IEEE Conference on Decision and Control*, pp. 5606–5611, IEEE, 2005.
- [28] T. Geyer, "Model predictive direct current control: Formulation of the stator current bounds and the concept of the switching horizon," *IEEE Industry Applications Magazine*, vol. 18, no. 2, pp. 47–59, 2011.
- [29] N. Oikonomou, C. Gutscher, P. Karamanakos, F. Kieferndorf, and T. Geyer, "Model predictive pulse pattern control for the five-level active neutral-point-clamped inverter," *IEEE Transactions on Industry Applications*, vol. 49, no. 6, pp. 2583–2592, 2013.
- [30] A. Bemporad, M. Morari, V. Dua, and E. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.
- [31] A. Alessio and A. Bemporad, "A survey on explicit model predictive control," in *Nonlinear model predictive control*, pp. 345–369, Springer, 2009.
- [32] P. Cortés, M. Kazmierkowski, R. Kennel, D. Quevedo, and J. Rodríguez, "Predictive control in power electronics and drives," *IEEE Transactions on industrial electronics*, vol. 55, no. 12, pp. 4312–4324, 2008.

- [33] P. Nguyen, “Lattice reduction algorithms: Theory and practice,” in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 2–6, Springer, 2011.
- [34] E. Agrell, T. Eriksson, and A. e. a. Vardy, “Closest point search in lattices,” *Information Theory, IEEE Transactions on*, vol. 48, no. 8, pp. 2201–2214, 2002.
- [35] S. Arora and B. Barak, *Computational complexity: a modern approach*. Cambridge University Press, 2009.
- [36] D. Micciancio, “The hardness of the closest vector problem with preprocessing,” *Information Theory, IEEE Transactions on*, vol. 47, pp. 1212–1215, 2001.
- [37] C. Lamy and J. Boutros, “On random rotations diversity and minimum mse decoding of lattices,” *Information Theory, IEEE Transactions on*, vol. 46, pp. 1584–1589, 2000.
- [38] O. Damen, A. Chkeif, and J. Belfiore, “Lattice code decoder for space-time codes,” *IEEE Communications letters*, vol. 4, no. 5, pp. 161–163, 2000.
- [39] T. Geyer and D. Quevedo, “Multistep finite control set model predictive control for power electronics,” *IEEE Transactions on Power Electronics*, vol. 29, pp. 6836–6846, 2014.
- [40] C. Bordons and C. Montero, “Basic principles of mpc for power converters: Bridging the gap between theory and practice,” *IEEE Industrial Electronics Magazine*, vol. 9, no. 3, pp. 31–43, 2015.
- [41] M. Pohst, “On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications,” *ACM SIGSAM Bulletin*, vol. 15, pp. 37–44, 1981.
- [42] R. Kannan, “Minkowski’s convex body theorem and integer programming,” *Mathematics of operation research*, vol. 12, no. 3, pp. 415–440, 1987.
- [43] U. Fincke and M. Pohst, “Improved methods for calculating vectors of short length in a lattice, including a complexity analysis,” *Mathematics of Computation*, vol. 44, no. 170, pp. 463–471, 1985.
- [44] C. Schnorr and M. Euchner, “Lattice basis reduction: Improved practical algorithms and solving subset sum problems,” *Mathematical programming*, vol. 66, no. 1-3, pp. 181–199, 1994.

- [45] M. Ajtai, R. Kumar, and D. Sivakumar, "A sieve algorithm for the shortest lattice vector problem," in *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pp. 601–610, ACM, 2001.
- [46] D. Micciancio and P. Voulgaris, "A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations," *SIAM Journal on Computing*, vol. 42, no. 3, pp. 1364–1391, 2013.
- [47] C. Garcia, D. Prett, and M. Morari, "Model predictive control: theory and practice - a survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.
- [48] P. Karamanakos, T. Geyer, T. Mouton, and R. Kennel, "Computationally efficient sphere decoding for long-horizon direct model predictive control," *Energy Conversion Congress and Exposition (ECCE)*, vol. IEEE, pp. 1–8, 2016.
- [49] L. Grune and A. Rantzer, "On the infinite horizon performance of receding horizon controllers," *IEEE Transactions on Automatic Control*, vol. 53, no. 9, pp. 2100–2111, 2008.
- [50] G. Golub and C. Van Loan, "Matrix computations, johns hopkins u," *Math. Sci., Johns Hopkins University Press, Baltimore, MD*, 1996.
- [51] R. Aguilera and D. Quevedo, "Predictive control of power converters: Designs with guaranteed performance," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 1, pp. 53–63, 2014.
- [52] W. Duesterhoeft, M. Schulz, and E. Clarke, "Determination of instantaneous currents and voltages by means of alpha, beta, and zero components," *Transactions of the American Institute of Electrical Engineers*, vol. 70, no. 2, pp. 1248–1255, 1951.
- [53] D. Zambra, C. Rech, and J. Pinheiro, "Comparison of neutral-point-clamped, symmetrical, and hybrid asymmetrical multilevel inverters," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 7, pp. 2297–2306, 2010.
- [54] M. Hasan, S. Mekhilef, and M. Ahmed, "Three-phase hybrid multilevel inverter with less power electronic components using space vector modulation," *IET Power Electronics*, vol. 7, no. 5, pp. 1256–1265, 2014.
- [55] P. Palanivel and S. Dash, "Analysis of thd and output voltage performance for cascaded multilevel inverter using carrier pulse width modulation techniques," *IET Power Electronics*, vol. 4, no. 8, pp. 951–958, 2011.



- [56] S. Kouro, M. Malinowski, K. Gopakumar, J. Pou, L. Franquelo, B. Wu, J. Rodriguez, M. Pérez, and J. Leon, "Recent advances and industrial applications of multilevel converters," *IEEE Transactions on industrial electronics*, vol. 57, no. 8, pp. 2553–2580, 2010.
- [57] A. Nabae, I. Takahashi, and H. Akagi, "A new neutral-point-clamped pwm inverter," *IEEE Transactions on industry applications*, no. 5, pp. 518–523, 1981.
- [58] J. Rodríguez, S. Bernet, B. Wu, J. Pontt, and S. Kouro, "Multilevel voltage-source-converter topologies for industrial medium-voltage drives," *IEEE Transactions on industrial electronics*, vol. 54, no. 6, pp. 2930–2945, 2007.
- [59] J. Holtz, "The representation of ac machine dynamics by complex signal flow graphs," *IEEE transactions on industrial electronics*, vol. 42, no. 3, pp. 263–271, 1995.
- [60] K. Hasse, "Zum dynamischen verhalten der asynchronmaschine bei betrieb mit variabler standerfrequenz und standerspannung," *ETZ-A Bd.*, vol. 89, pp. 387–391, 1968.
- [61] I. Takahashi and Y. Ohmori, "High-performance direct torque control of an induction motor," *IEEE Transactions on Industry Applications*, vol. 25, no. 2, pp. 257–264, 1989.
- [62] J. Rodríguez, R. Kennel, J. Espinoza, M. Trincado, C. Silva, and C. Rojas, "High-performance control strategies for electrical drives: An experimental assessment," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 2, pp. 812–820, 2011.
- [63] R. Kennel and A. Linder, "Predictive control for electrical drives—a survey," *Korea-Germany Advanced Power Electronics Symposium 2001*, pp. 39–43, 2001.
- [64] A. Linder and R. Kennel, "Model predictive control for electrical drives," in *2005 IEEE 36th Power Electronics Specialists Conference*, pp. 1793–1799, IEEE, 2005.
- [65] S. Mariéthoz, A. Domahidi, and M. Morari, "Sensorless explicit model predictive control of permanent magnet synchronous motors," in *2009 IEEE International Electric Machines and Drives Conference*, pp. 1250–1257, IEEE, 2009.
- [66] T. Geyer, "Generalized model predictive direct torque control: Long prediction horizons and minimization of switching losses," in *Proceedings of the 48th IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pp. 6799–6804, IEEE, 2009.

- [67] T. Geyer, G. Papafotiou, and M. Morari, "Model predictive direct torque control - part i: Concept, algorithm, and analysis," *IEEE transactions on industrial electronics*, vol. 56, no. 6, pp. 1894–1905, 2008.
- [68] J. Scoltock, T. Geyer, and U. Madawala, "A comparison of model predictive control schemes for mv induction motor drives," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 2, pp. 909–919, 2012.
- [69] T. Geyer, *Low complexity model predictive control in power electronics and power systems*. Cuvillier Verlag, 2005.
- [70] IEEE, "Standard 519-1992: Recommended practices and requirements for harmonic control in electrical power systems," *New York, NY, USA*, 1992.
- [71] B. Hassibi and H. Vikalo, "On the expected complexity of integer least-squares problems," in *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, vol. 2, pp. II–1497, IEEE, 2002.
- [72] S. Galbraith, *Mathematics of public key cryptography*. Cambridge University Press, 2012.
- [73] H. Coxeter and S. Macdonald, *Regular polytopes*. Courier Corporation, 1973.
- [74] M. Joye and M. Tunstall, *Fault analysis in cryptography*, vol. 147. Springer, 2012.
- [75] D. Micciancio and S. Goldwasser, *Complexity of lattice problems: a cryptographic perspective*, vol. 671. Springer Science & Business Media, 2012.
- [76] D. Micciancio, "Lattice algorithms and applications: Point lattices." <https://cseweb.ucsd.edu/classes/sp14/cse206A-a/lec1.pdf>.
- [77] M. Krein and D. Milman, "On extreme points of regular convex sets," *Studia Mathematica*, vol. 9, pp. 133–138, 1940.
- [78] 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, 2005.
- [79] D. Shreiner, G. Sellers, J. Kessenich, and B. Licea-Kane, *OpenGL programming guide: The Official guide to learning OpenGL, version 4.3*. Addison-Wesley, 2013.
- [80] R. Rockafellar, *Convex analysis*. No. 28, Princeton university press, 1970.
- [81] J. Cassels, *An introduction to the geometry of numbers*. Springer Science & Business Media, 2012.

- [82] F. Aurenhammer, "Voronoi diagrams - a survey of a fundamental geometric data structure," *ACM Computing Surveys (CSUR)*, vol. 23, no. 3, pp. 345–405, 1991.
- [83] G. Voronoi, "Nouvelles applications des parametres continus a la theorie des formes quadratiques," *Math.*, vol. 133, pp. 97–178, 1908.
- [84] J. Conway and N. Sloane, *Sphere packings, lattices and groups*, vol. 290. Springer Science & Business Media, 2013.
- [85] P. Gruber, "Geometry of numbers," in *Handbook of convex geometry*, pp. 739–763, Elsevier, 1993.
- [86] E. Viterbo and E. Biglieri, "Computing the voronoi cell of a lattice: the diamond-cutting algorithm," *IEEE transactions on information theory*, vol. 42, no. 1, pp. 161–171, 1996.
- [87] P. Gruber and C. Lekkerkerker, *Geometry of numbers*. North-Holland, 1987.
- [88] E. Schulte, "Tilings," in *Handbook of Convex Geometry*, pp. 899–932, Elsevier, 1993.
- [89] J. Blömer and J. Seifert, "On the complexity of computing short linearly independent vectors and short bases in a lattice," in *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pp. 711–720, ACM, 1999.
- [90] D. Micciancio, "The hardness of the closest vector problem with preprocessing," *IEEE Transactions on Information Theory*, vol. 47, no. 3, pp. 1212–1215, 2001.
- [91] D. Micciancio, "The shortest vector in a lattice is hard to approximate to within some constant," *SIAM journal on Computing*, vol. 30, no. 6, pp. 2008–2035, 2001.
- [92] S. Khot, "Hardness of approximating the shortest vector problem in lattices," *Journal of the ACM (JACM)*, vol. 52, no. 5, pp. 789–808, 2005.
- [93] T. Dorfling, "Practical implementation of long-horizon direct model predictive control," Master's thesis, Stellenbosch: Stellenbosch University, 2018.
- [94] T. Dorfling, H. du Toit Mouton, T. Geyer, and P. Karamanakos, "Long-horizon finite-control-set model predictive control with nonrecursive sphere decoding on an fpga," *IEEE Transactions on Power Electronics*, vol. 35, no. 7, pp. 7520–7531, 2019.
- [95] R. Baidya, *Multistep Model Predictive Control for Power Electronics and Electrical Drives*. PhD thesis, Electrical Engineering and Telecommunications, University of New South Wales, Sydney, Australia, 2018.

- [96] G. Golub and C. Van Loan, *Matrix computations*, vol. 3. JHU press, 2012.
- [97] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
- [98] L. Babai, “On lovász lattice reduction and the nearest lattice point problem,” *Combinatorica*, vol. 6, no. 1, pp. 1–13, 1986.
- [99] P. Acuna, C. Rojas, R. Baidya, R. Aguilera, and J. Fletcher, “On the impact of transients on multistep model predictive control for medium-voltage drives,” *IEEE Transactions on Power Electronics*, vol. 34, no. 9, pp. 8342–8355, 2019.
- [100] R. Baidya, R. Aguilera, P. Acuña, T. Geyer, R. Delgado, D. Quevedo, and H. du Toit Mouton, “Enabling multistep model predictive control for transient operation of power converters,” *IEEE Open Journal of the Industrial Electronics Society*, vol. 1, pp. 284–297, 2020.
- [101] N. Sommer, M. Feder, and O. Shalvi, “Finding the closest lattice point by iterative slicing,” *SIAM Journal on Discrete Mathematics*, vol. 23, no. 2, pp. 715–731, 2009.
- [102] D. Luenberger, *Optimization by vector space methods*. John Wiley & Sons, 1997.
- [103] J. Raath, T. Mouton, and T. Geyer, “On the micciancio-voulgaris algorithm to solve the long-horizon direct mpc optimization problem,” in *Predictive Control of Electrical Drives and Power Electronics (PRECEDE), 2017 IEEE International Symposium on*, pp. 95–100, IEEE, 2017.
- [104] D. Quevedo, J. De Doná, and G. Goodwin, “Receding horizon linear quadratic control with finite input constraint sets,” *IFAC Proceedings Volumes*, vol. 35, no. 1, pp. 183–188, 2002.
- [105] R. Baidya, R. Aguilera, P. Acuna, R. Delgado, T. Geyer, D. Quevedo, and T. Mouton, “Fast multistep finite control set model predictive control for transient operation of power converters,” in *Industrial Electronics Society, IECON 2016-42nd Annual Conference of the IEEE*, pp. 5039–5045, IEEE, 2016.
- [106] P. Karamanakos, T. Geyer, and R. Aguilera, “Computationally efficient long-horizon direct model predictive control for transient operation,” in *Energy Conversion Congress and Exposition (ECCE), 2017 IEEE*, pp. 4642–4649, IEEE, 2017.
- [107] G. Strang, *Linear Algebra and its Applications*, vol. 3. Harcourt Brace Jovanovich, 1988.
- [108] D. Nykamp, “How linear transformations map parallelograms and parallelepipeds.” [http://mathinsight.org/linear\\_transformations\\_map\\_parallelograms\\_parallelepipeds](http://mathinsight.org/linear_transformations_map_parallelograms_parallelepipeds).

- [109] L. Nirenberg, *Topics in nonlinear functional analysis*, vol. 6. American Mathematical Soc., 1974.
- [110] W. Cheney and D. Kincaid, “Linear algebra: Theory and applications,” *The Australian Mathematical Society*, vol. 110, 2009.
- [111] M. Dutour, A. Schürmann, and F. Vallentin, “Complexity and algorithms for computing voronoi cells of lattices,” *Mathematics of computation*, vol. 78, no. 267, pp. 1713–1731, 2009.
- [112] V. Guruswami, D. Micciancio, and O. Regev, “The complexity of the covering radius problem,” *Computational Complexity*, vol. 14, no. 2, pp. 90–121, 2005.
- [113] E. Viterbo and E. Biglieri, “Computing the voronoi cell of a lattice: the diamond-cutting algorithm,” *IEEE transactions on information theory*, vol. 42, no. 1, pp. 161–171, 1996.
- [114] M. Herceg, M. Kvasnica, C. Jones, and M. Morari, “Multi-Parametric Toolbox 3.0,” in *Proc. of the European Control Conference*, (Zürich, Switzerland), pp. 502–510, July 17–19 2013. <http://control.ee.ethz.ch/~mpt>.
- [115] D. Watkins, *Fundamentals of matrix computations*, vol. 64. John Wiley & Sons, 2004.
- [116] T. Laczynski and A. Mertens, “Predictive stator current control for medium voltage drives with lc filters,” *IEEE Transactions on Power Electronics*, vol. 24, no. 11, pp. 2427–2435, 2009.
- [117] J. Steinke, “Use of an lc filter to achieve a motor-friendly performance of the pwm voltage source inverter,” *IEEE transactions on Energy Conversion*, vol. 14, no. 3, pp. 649–654, 1999.
- [118] B. Schmitt and R. Sommer, “Retrofit of fixed speed induction motors with medium voltage drive converters using npc three-level inverter high-voltage igbt based topology,” in *ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No. 01TH8570)*, vol. 2, pp. 746–751, IEEE, 2001.
- [119] K. Mohanty, “A closed loop observer for rotor flux estimation in induction machines,” *49th Annual session of Orissa Engineering congress*, pp. 6–12, 2004.
- [120] P. Cortes, J. Rodriguez, C. Silva, and A. Flores, “Delay compensation in model predictive current control of a three-phase inverter,” *IEEE Transactions on Industrial Electronics*, vol. 59, no. 2, pp. 1323–1325, 2011.

- [121] F. Kieferndorf, P. Karamanakos, P. Bader, N. Oikonomou, and T. Geyer, “Model predictive control of the internal voltages of a five-level active neutral point clamped converter,” in *2012 IEEE Energy Conversion Congress and Exposition (ECCE)*, pp. 1676–1683, IEEE, 2012.
- [122] J. He, G. Sizov, P. Zhang, and N. Demerdash, “A review of mitigation methods for over-voltage in long-cable-fed pwm ac drives,” in *2011 IEEE Energy Conversion Congress and Exposition*, pp. 2160–2166, IEEE, 2011.
- [123] J. Pontt, J. Rodriguez, S. Kouro, C. Silva, H. Farias, and M. Rotella, “Output sinus filter for medium voltage drive with direct torque control,” in *Fourtieth IAS Annual Meeting. Conference Record of the 2005 Industry Applications Conference, 2005.*, vol. 1, pp. 204–209, IEEE, 2005.
- [124] J. Bélanger, P. Venne, and J. Paquin, “The what, where and why of real-time simulation,” *Planet RT*, vol. 1, no. 1, pp. 25–29, 2010.
- [125] C. Dufour, C. Andrade, and J. Bélanger, “Real-time simulation technologies in education: a link to modern engineering methods and practices,” 2010.
- [126] OPAL-RT, “Op5600v2 system description.” <https://www.opal-rt.com/simulator-platform-op5600>, 2019.
- [127] INTEL, “Intel processors.” <https://www.intel.com/content/www/us/en/products/processors/core/i5-processors/i5-6200u.html>.
- [128] N. Johnson, *Geometries and Transformations*. Cambridge University Press, 2018.
- [129] J. Dattorro, “Convex optimization and euclidean distance geometry,” *Meboo, Palo Alto*, 2005.
- [130] V. Bengtsson, “Orthogonal decomposition.” <https://mathworld.wolfram.com>.