Generalized Model Predictive Pulse Pattern Control Based on Small-Signal Modelling—Part 2: Implementation and Analysis

Tinus Dorfling, Hendrik du Toit Mouton, *Member, IEEE*, and Tobias Geyer, *Fellow, IEEE*

***Abstract*—A newly-proposed model predictive pulse pattern controller that is applicable to higher-order converter systems is implemented on a low-cost field-programmable gate array. It is shown how the computations of the underlying quadratic program of the control algorithm can be reduced, thus enabling a resource-efficient implementation. The implemented control algorithm is evaluated via hardware-in-the-loop simulations. The results show that the proposed controller achieves a short settling time during transients and the superior harmonic performance of optimized pulse patterns during steady-state operation.**

***Index Terms*—Optimized pulse patterns, model predictive control, field-programmable gate arrays**

I. Introduction

Optimized pulse patterns (OPPs) are a pulse-width modulation method in which the switching patterns are computed offline by minimizing an objective function [1], [2]. OPPs are known for their superior harmonic performance, significantly outperforming well-known carrier-based pulse-width modulation at low pulse numbers (that is, at low switching frequency to fundamental frequency ratios), see [3] and [4, Chapter 13.2]. This makes OPPs particularity well-suited to high-power applications, such as medium-voltage drives. However, fast closed-loop control of an OPP-modulated higher-order converter system, such as a grid-connected converter with an LC filter, is difficult to achieve, and is a largely unresolved problem.

In the first part of this paper [5], a generalization of the model predictive pulse pattern controller [6] was proposed to solve the high-bandwidth control problem underlying higher-order OPP-modulated converters. The controller—known as the *small-signal controller*—regulates the state vector $\boldsymbol{x}$ of a converter system along its (optimal) steady-state trajectory $\boldsymbol{x}^*$ by modifying the switching instants of the nominal pulse pattern $\boldsymbol{u}^*_{abc}$, resulting in a modified pulse pattern $\boldsymbol{u}_{abc}$. It was shown that the strengths of impulses can be used to represent the areas that are added or removed to a pulse pattern (in other words, the modifications) as

$$\lambda_{p,i} = -\Delta t_{p,i} \Delta u_{p,i}, \tag{1}$$

where $\Delta t_{p,i} = t_{p,i} - t^*_{p,i}$ is a time modification, with $t_{p,i}$ and $t^*_{p,i}$ being the modified and nominal switching instants, respectively; and where $\Delta u_{p,i} = u_{p,i} - u_{p,i-1}$ is a switching transition, with $u_{p,i}, u_{p,i-1} \in \mathbb{Z}$. Refer to Fig. 1 for an illustration. The index $p \in \{a, b, c\}$ denotes a particular phase, and the index i denotes the ith switching transition in that phase. Note that the time modifications are encoded in the strengths of the impulses. Thanks to the use of strengths of impulses, the underlying optimization problem of the control algorithm is the quadratic program (QP) [7, Chapter 16]

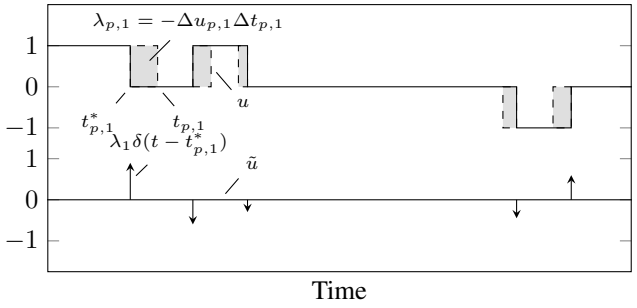


Fig. 1: Impulses represent areas that are added or removed from a pulse pattern. One phase of a three-level pulse pattern is shown.

$$\boldsymbol{\Lambda}_{\text{opt}} = \arg\min_{\boldsymbol{\Lambda}} \quad \tfrac{1}{2}\boldsymbol{\Lambda}^{\text{T}}\boldsymbol{H}\boldsymbol{\Lambda} + \boldsymbol{c}^{\text{T}}\boldsymbol{\Lambda} \tag{2a}$$

$$\text{subject to } \boldsymbol{A}\boldsymbol{\Lambda} \leq \boldsymbol{b}. \tag{2b}$$

The decision vector $\boldsymbol{\Lambda} \in \mathbb{R}^{n_{\text{sw}}}$, where $n_{\text{sw}} = n_a + n_b + n_c$ is the number of switching transition that fall within the prediction horizon T_p, is known as the *strength vector*; it is defined as

$$\boldsymbol{\Lambda} = [\lambda_{a,1} \ \cdots \ \lambda_{a,n_a} \ \lambda_{b,1} \ \cdots \ \lambda_{b,n_b} \ \lambda_{c,1} \ \cdots \ \lambda_{c,n_c}]^{\text{T}}. \tag{3}$$

The matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$ of the inequality constraint (2b) are shown in Appendix B. Once the QP has been solved, the optimal strength vector $\boldsymbol{\Lambda}_{\text{opt}}$ is obtained. By re-writing (1), the (optimal) impulse strengths can be translated to the time modifications

$$\Delta t_{\text{opt},p,i} = -\frac{\lambda_{\text{opt},p,i}}{\Delta u_{p,i}}, \tag{4}$$

which are then added to the nominal switching times to obtain the (optimal) modified switching instants

$$t_{\text{opt},p,i} = t^*_{p,i} + \Delta t_{\text{opt},p,i}. \tag{5}$$

This paper presents an efficient implementation of the small-signal controller from [5]. To the authors' knowledge, this is the first time that a (real-time) control algorithm for

T. Dorfling and H dT. Mouton are with Department of Electrical and Electronic Engineer, Stellenbosch University, Stellenbosch 7599; South Africa; e-mail: martinusdorfling@protonmail.com, dtmouton@sun.ac.za

T. Geyer is with ABB System Drives, Turgi 5300, Switzerland; e-mail: t.geyer@ieee.org

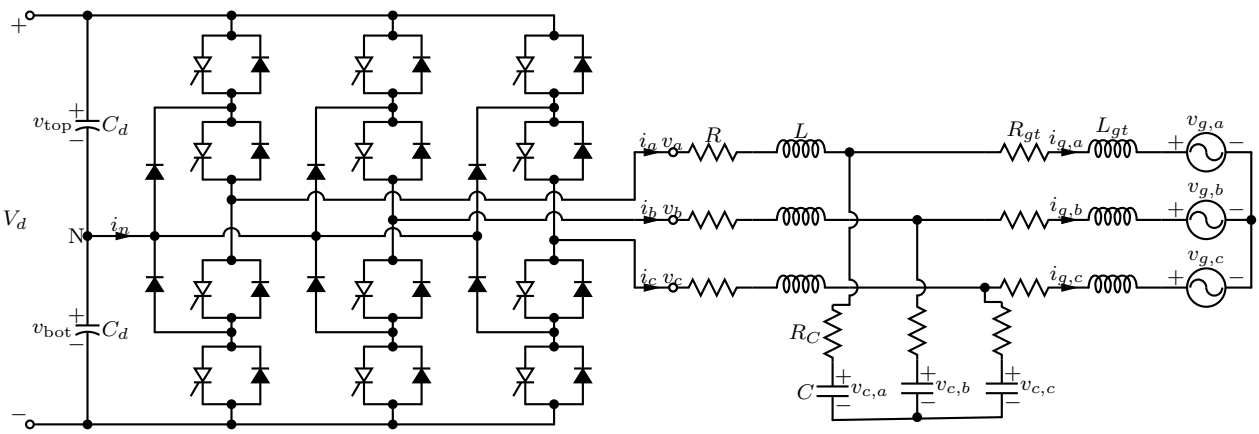


Fig. 2: Grid-connected neutral-point-clamped converter with an LC filter. The resistances and inductances of the grid and transformer are lumped together as $R_{gt} = R_g + R_t$ and $L_{gt} = L_g + L_t$, respectively. Note that the resistors represent the equivalent series resistances of the filter inductor and capacitor, as well as the winding resistance of the transformer.

an OPP-modulated higher-order converter system has been implemented on a real-time embedded system. Specific attention is given to formulating and solving the QP of (2), which is the part of the control algorithm with the highest computational burden. Other aspects of the control algorithm are not discussed, and the reader is referred to [5, Section V-H] for more details. Specifically, this paper explains how to (efficiently) calculate the quadratic and linear terms in the objective function (that is, the QP formulation), and how to solve the QP using the gradient projection method. The former is explained in Section II, and the latter in Section III. This is followed by a brief discussion of the implementation of the control algorithm on a low-cost field-programmable gate array (FPGA) in Section IV. The control algorithm is then verified in real-time via a hardware-in-the-loop (HIL) simulation in Section V.

As a case study, a neutral-point-clamped (NPC) converter connected to the grid via an LC filter is used, as shown in Fig. 2. For more information on the converter system, refer to [5, Section III-B].

II. QP Formulation

The first step regarding the QP is to calculate the Hessian $\boldsymbol{H} \in \mathbb{R}^{n_{\mathrm{sw}} \times n_{\mathrm{sw}}}$ and vector $\boldsymbol{c} \in \mathbb{R}^{n_{\mathrm{sw}}}$ of the quadratic objective function. It is important to note that $\boldsymbol{H}$ and $\boldsymbol{c}$ have to be calculated at *every* sampling instant of the control algorithm, as they are not constant.

Recall from [5, (51)] that the Hessian is defined as $\boldsymbol{H} = \boldsymbol{V} + \boldsymbol{R}$. The matrix $\boldsymbol{R}$ is simply a diagonal matrix that penalizes the control effort. According to [5, (45)], the (i', j')th component of $\boldsymbol{V}$ is defined as

$$\boldsymbol{V}_{(i',j')} = \boldsymbol{G}_{p_1}^{\mathrm{T}} e^{\boldsymbol{F}^{\mathrm{T}}(t_{ij}^* - t_{p_1,i}^*)} \boldsymbol{\Xi}(T_p - t_{ij}^*) e^{\boldsymbol{F}(t_{ij}^* - t_{p_2,j}^*)} \boldsymbol{G}_{p_2}, \tag{6}$$

where $t_{ij}^* = \max\{t_{p_1,i}^*, t_{p_2,j}^*\}$, and

$$\boldsymbol{\Xi}(\Delta T) = \int_0^{\Delta T} e^{\boldsymbol{F}^{\mathrm{T}} t} \boldsymbol{Q} e^{\boldsymbol{F} t} \, \mathrm{d}t.$$

The index i' corresponds to phase $p_1 \in \{a, b, c\}$ and the ith switching transition in that phase. The index j' is defined accordingly based on p_2. Refer to [5, (47)] on how to solve $\boldsymbol{\Xi}$. Furthermore, the i'th component of $\boldsymbol{c}$ is

$$\boldsymbol{c}_{i'}^{\mathrm{T}} = \tilde{\boldsymbol{x}}_0^{\mathrm{T}} e^{\boldsymbol{F}^{\mathrm{T}} t_{p,i}^*} \boldsymbol{\Xi}(T_p - t_{p,i}^*) \boldsymbol{G}_p. \tag{7}$$

Recall that $\boldsymbol{F} \in \mathbb{R}^{6\times 6}$ and $\boldsymbol{G} \in \mathbb{R}^{6\times 3}$ are the system and input matrices, respectively, of the state-space representation of the grid-connected converter system. From [5, Appendix A], these matrices are given as[1]

$$\begin{aligned} \boldsymbol{F} &= \begin{bmatrix} -\frac{R+R_C}{L}\mathbf{I}_2 & \frac{R_C}{L}\mathbf{I}_2 & -\frac{1}{L}\mathbf{I}_2 \\ \frac{R_C}{L_{gt}}\mathbf{I}_2 & -\frac{R_{gt}+R_C}{L_{gt}}\mathbf{I}_2 & \frac{1}{L_{gt}}\mathbf{I}_2 \\ \frac{1}{C}\mathbf{I}_2 & -\frac{1}{C}\mathbf{I}_2 & \mathbf{0}_{2\times 2} \end{bmatrix} \\ \boldsymbol{G} &= \frac{V_d}{2L} \begin{bmatrix} \mathbf{I}_2 \\ \mathbf{0}_{2\times 2} \\ \mathbf{0}_{2\times 2} \end{bmatrix} \boldsymbol{K}, \end{aligned} \tag{8}$$

for a state vector that is defined as

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

Furthermore, $\boldsymbol{G}_a = \boldsymbol{G}[1\ 0\ 0]^{\mathrm{T}}$, $\boldsymbol{G}_b = \boldsymbol{G}[0\ 1\ 0]^{\mathrm{T}}$, and $\boldsymbol{G}_c = \boldsymbol{G}[0\ 0\ 1]^{\mathrm{T}}$ are vectors that were defined in [5, Section V-B].

The problem structure and the load characteristics can be exploited when calculating (6) and (7), as shown in Appendix A. This reduces the computational burden. Note that the computations regarding the matrix exponentials are not discussed; it is assumed that the matrix exponentials are calculated offline and stored in lookup tables.

III. QP Solver

Once the Hessian $\boldsymbol{H}$ and vector $\boldsymbol{c}$ have been calculated, the QP of (2) can be solved using the gradient projection method. The kth iteration of the method, with stepsize s and gradient $\nabla f(\boldsymbol{\Lambda}_k) = \boldsymbol{H}\boldsymbol{\Lambda}_k + \boldsymbol{c}$, is

$$\boldsymbol{\Lambda}_{k+1} = \pi_{\mathcal{Z}}\left(\boldsymbol{\Lambda}_k - s\left(\boldsymbol{H}\boldsymbol{\Lambda}_k + \boldsymbol{c}\right)\right) \tag{9}$$

[1]The matrix $\boldsymbol{K} = \frac{2}{3}\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix}$ denotes the (reduced) Clarke transformation matrix. Here, $\mathbf{I}_2$ denotes the identity matrix of dimension two, and $\mathbf{0}_{2\times 2}$ is the 2×2 zero matrix.

where the feasible set is $\mathcal{Z} = \{\boldsymbol{z} : \boldsymbol{A}\boldsymbol{z} \leq \boldsymbol{b}\}$. The operator $\pi_{\mathcal{Z}}(\cdot)$ projects the result onto the set $\mathcal{Z}$. In this paper, a fixed stepsize of $s = \frac{1}{L_c}$ is used for the gradient method, where $L_c = \|\boldsymbol{H}\|_2$ is known as the tight Lipschitz constant. Here, $\|\boldsymbol{H}\|_2$ denotes the induced 2-norm of the Hessian. For further reading on the gradient projection method, refer to [8, Section 3.3] and [9, Section 9.4].

In this section, it is shown how the stepsize s can be efficiently calculated; this is required at each sampling instant, since $\boldsymbol{H}$ is time-varying. Once the stepsize s has been calculated, it is then shown how the projection operation onto the set $\mathcal{Z}$ can be efficiently realized.

A. Efficiently Determining the Stepsize

In order to calculate the stepsize

$$s = \frac{1}{L_c},$$

two operations are required: first, the Lipschitz constant L_c has to be calculated, which is then followed by calculating its reciprocal in order to determine the stepsize s.

1) Efficiently Overestimating the Lipschitz Constant: Consider the Lipschitz constant L_c, which requires the evaluation of the (induced) 2-norm of the Hessian $\boldsymbol{H}$. Evaluating the 2-norm of a matrix is computationally expensive, requiring an iterative method. Fortunately, a useful relation from [10, Section 2.3.3] states that the 2-norm of a matrix can be overestimated as

$$\|\boldsymbol{H}\|_2 \leq \sqrt{\|\boldsymbol{H}\|_1 \|\boldsymbol{H}\|_\infty},$$

and since the Hessian $\boldsymbol{H}$ is symmetric, the relation simplifies to

$$\|\boldsymbol{H}\|_2 \leq \|\boldsymbol{H}\|_1 = \|\boldsymbol{H}\|_\infty.$$

Evaluating the infinity-norm (or 1-norm) of a matrix is computationally very cheap, as it is simply

$$\|\boldsymbol{H}\|_\infty = \max_{1 \leq i \leq n_{\mathrm{sw}}} \sum_{j=1}^{n_{\mathrm{sw}}} \left| \boldsymbol{H}_{(i,j)} \right|, \tag{10}$$

which is the maximum of the absolute value of the matrix entries summed up in each row; no multiplications are required. Denote with $\bar{L_c} = \|\boldsymbol{H}\|_\infty$ the overestimated (tight) Lipschitz constant. An overestimated (tight) Lipschitz constant $\bar{L_c}$ results in an underestimated stepsize s.

2) Calculating a Reciprocal: To calculate the stepsize $s = \frac{1}{\bar{L_c}}$, division needs to be implemented on the FPGA, since hardware description languages typically do not have a division primitive. A widely-used numerical method that can be used to realize division is Newton's method [11].

In a first step, the Lipschitz constant $\bar{L_c}$ is normalized to the interval $[0.5, 1]$ (this is common practice, and the reasoning for this will soon become apparent). The normalization can be easily achieved by using simple bit shifts, which are equivalent to dividing or multiplying by powers of two. Denote with $D = \bar{L_c} 2^m \in [0.5, 1]$ the normalized Lipschitz constant, where $m \in \mathbb{Z}$ is the number of bit shifts required to scale $\bar{L_c}$ to the interval $[0.5, 1]$. Then,

$$\frac{1}{D}$$

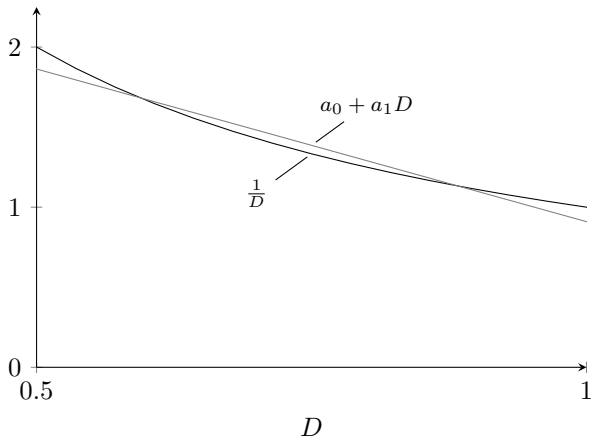


Fig. 3: Linear least-square approximation of $\frac{1}{D}$ in the region $D \in [0.5, 1]$. The coefficients are $a_0 = 2.8162$ and $a_1 = -1.9066$.

can be approximated with z by finding the root of

$$f(z) = \frac{1}{z} - D$$

by using Newton's method, where the kth iteration has the form [11]

$$z_{k+1} = z_k + z_k(1 - D z_k). \tag{11}$$

Once Newton's method has terminated, the final iteration z_{opt} (which approximates $\frac{1}{D}$) is denormalized to find the stepsize as $s = z_{\mathrm{opt}} 2^m$.

Note that in order for the method to converge, the initial solution z_0 must fall within the interval $(0, \frac{2}{D})$. Recall that Newton's method has quadratic convergence, meaning the error reduces by $\epsilon_{i+1} = \epsilon_i^2$ at subsequent iterations. Thus, finding an initial iterate that is close to the solution $\frac{1}{D}$ *significantly* increases the convergence, since a small initial error (given by $\epsilon_0 = 1 - D z_0$) will rapidly diminish at subsequent iterations. To generate an initial solution z_0, consider the linear least-squares approximation problem

$$\min_{a_0, a_1} \int_{0.5}^{1} \left(\frac{1}{D} - (a_0 + a_1 D) \right)^2 \mathrm{d}D. \tag{12}$$

The (linear) least-squares problem of (12) involves calculating the coefficients a_0 and a_1 so that the first-order polynomial $a_0 + a_1 D$ best approximates $\frac{1}{D}$ (in the least-square sense) in the region $[0.5, 1]$. According to [12, Section 8.2], the coefficients a_0 and a_1 are the (unique) solutions to the system of linear equations

$$a_0 \int_{0.5}^{1} 1 \mathrm{d}D + a_1 \int_{0.5}^{1} D \mathrm{d}D = \int_{0.5}^{1} \frac{1}{D} \mathrm{d}D$$
$$a_0 \int_{0.5}^{1} D \mathrm{d}D + a_1 \int_{0.5}^{1} D^2 \mathrm{d}D = \int_{0.5}^{1} 1 \mathrm{d}D.$$

After solving the trivial integrals, the solutions to the system of linear equations are $a_0 = 2.8162$ and $a_1 = -1.9066$. The linear least-square approximation is shown in Fig. 3. The initial iterate z_0 of Newton's method is set equal to the linear (least-squares) approximation of $\frac{1}{D}$, $z_0 = a_0 + a_1 D$. Note that these (static) coefficients always result in the best linear approximation (in the least-square sense) for the initial iterate

z_0, since D is *always* in the interval $[0.5, 1]$. With the initial iterate $z_0 = a_0 + a_1 D$, only two to three iterations of (11) are (typically) required for sufficient accuracy.

B. Efficient Gradient Projection Method

As seen from (9), the gradient projection method involves two steps. The first step, taking the (unconstrained) step

$$\boldsymbol{\Lambda}_k - \frac{1}{\bar{L}_c}\left(\boldsymbol{H}\boldsymbol{\Lambda}_k + \boldsymbol{c}\right) \tag{13}$$

is trivial to realize. The second step, projecting the unconstrained step onto the set $\mathcal{Z}$, requires special attention to efficiently implement.

1) Transforming the Decision Variable for an Efficient Projection: In a first step, the decision variable is transformed to the modified switching instants which enables an efficient projection onto the feasible set. From [5, Section V-D], it is shown that the constraints $\mathcal{Z} = \{\boldsymbol{z} : \boldsymbol{A}\boldsymbol{z} \leq \boldsymbol{b}\}$ translate to

$$0 \leq t_{p,1} \leq t_{p,2} \leq \cdots \leq t_{p,n_p} \leq T_p \tag{14}$$

for all $p \in \{a, b, c\}$. The impulse strengths (which are the elements in the strength vector $\boldsymbol{\Lambda}$) can be written in terms of the modified switching instants as

$$t_{p,i} = t^*_{p,i} + \frac{\lambda_{p,i}}{\Delta u_{p,i}} \tag{15}$$

(recall that $\Delta u_{p,i} \in \{-1, 1\}$, no division is required). Once an unconstrained step has been taken according to (13), the impulse strengths are transformed to modified switching instants according to (15) before the result is projected.

Next, it is shown how to efficiently project onto the set of (14). Note the constraints of a phase are independent of the constraints in another phase; the projection of each phase can be considered separately. Thus, only the projection of a single phase is considered from here on in. Denote with $\boldsymbol{t}_p \in \mathbb{R}^{n_p}$ the vector of modified switching instants that is to be projected.

2) Efficient Projection onto a Truncated Monotone Cone: Although most of the results in this section follows from [13], they are repeated here for convenience.

The constraints of (14) form a so-called *truncated monotone cone*, which in its general form is

$$\bar{\mathbb{K}} = \{\boldsymbol{z} : \underline{z} \leq z_1 \leq z_2 \leq \cdots \leq z_{n_z} \leq \bar{z}\}, \tag{16}$$

where $\underline{z}$ and $\bar{z}$ are the lower and upper bounds, respectively.[2] As noted in [13], a truncated monotone cone can be written as the intersection between a (convex) monotone cone and a box,

$$\bar{\mathbb{K}} = \mathbb{K} \cap \mathbb{B},$$

where

$$\mathbb{K} = \{\boldsymbol{z} : z_1 \leq z_2 \leq \cdots \leq z_{n_z}\}$$
$$\mathbb{B} = \{\boldsymbol{z} : \underline{z} \leq z_i \leq \bar{z} \text{ for } i = 1, 2, \ldots, n_z\}$$

are the monotone cone and box, respectively. It is shown according to [13, Theorem 1] that the projection of $\boldsymbol{t}_p$ onto a truncated monotone cone is equivalent to first projecting it onto the monotone cone and then onto the box,

$$\pi_{\bar{\mathbb{K}}}(\boldsymbol{t}_p) = \pi_{\mathbb{B}}(\pi_{\mathbb{K}}(\boldsymbol{t}_p)). \tag{17}$$

The second projection, onto the box $\mathbb{B}$, is extremely simple to realize: the projection of the ith component of $\boldsymbol{\xi}$ is simply

$$[\pi_{\mathbb{B}}(\boldsymbol{\xi})]_i = \min\{\max\{\xi_i, \underline{z}\}, \bar{z}\}. \tag{18}$$

The first projection onto the monotone cone $\mathbb{K}$ is significantly more complex to realize, and is explained next.

Note that a monotone cone can be defined in matrix form as $\mathbb{K} = \{\boldsymbol{z} : \boldsymbol{C}\boldsymbol{z} \leq \boldsymbol{0}_{n_z - 1}\}$, where $\boldsymbol{C} \in \mathbb{R}^{(n_z-1)\times n_z}$ is a first-order difference matrix,

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

Although no closed-form expression exists for the projection onto a monotone cone, an exact solution can be iteratively obtained (see [14]). However, the exact solution is computationally expensive, requiring pseudoinverses of matrices, and is not pursued in this paper. Fortunately, [13] proposed a computationally efficient approximation that requires no (pseudo)inverses of matrices; due to this advantage, the approach of [13] is thus adopted in this work. As shown in [13, (25a)], the approximated projection follows as

$$\pi_{\mathbb{K}}(\boldsymbol{t}_p) = \boldsymbol{t}_p - \boldsymbol{C}^{\mathrm{T}}\boldsymbol{\eta}_{\mathrm{opt}} \tag{19}$$

with

$$\boldsymbol{\eta}_{\mathrm{opt}} = \arg\min_{\boldsymbol{\eta} \geq \boldsymbol{0}} \tfrac{1}{2}\boldsymbol{\eta}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{C}^{\mathrm{T}}\boldsymbol{\eta} - (\boldsymbol{C}\boldsymbol{y})^{\mathrm{T}}\boldsymbol{\eta}, \tag{20}$$

where $\boldsymbol{\eta} \in \mathbb{R}^{n_p - 1}$ is the Lagrange multiplier. In order to solve (20) to a certain accuracy, the gradient projection method can be used; refer to the gradient projection that is employed to solve (20) as the *inner* gradient method, whereas (9) is referred to as the *outer* gradient method. It is shown with [13, Proposition 2] that the largest L_d and smallest μ_d eigenvalues of the (dual) Hessian $\boldsymbol{C}\boldsymbol{C}^{\mathrm{T}} \in \mathbb{R}^{(n_p-1)\times(n_p-1)}$, which is a second-order difference matrix,

$$\boldsymbol{C}\boldsymbol{C}^{\mathrm{T}} = \begin{bmatrix} 2 & -1 & 0 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 & 2 & -1 \\ 0 & 0 & \cdots & 0 & -1 & 2 \end{bmatrix},$$

always satisfy

$$L_d + u_d = 4.$$

This results in an optimal (fixed) step size of $s = \frac{2}{L_d + u_d} = \frac{1}{2}$ for the inner gradient method; thus, the kth (inner) iteration is

$$\boldsymbol{\eta}_{k+1} = \max\left\{\boldsymbol{0}_{n_p-1}, \boldsymbol{\eta}_k - \tfrac{1}{2}(\boldsymbol{C}\boldsymbol{C}^{\mathrm{T}}\boldsymbol{\eta}_k - \boldsymbol{C}\boldsymbol{t}_p)\right\}. \tag{21}$$

Importantly, an iteration of (21) only requires additions, subtractions, and bit shifts; no multiplications are required. It is noted in [13] that if warmstarting is employed from the

[2] In the case of (14), the lower and upper bounds are $\underline{z} = 0$ and $\bar{z} = T_p$, respectively.

last iterate $\boldsymbol{\eta}_{k+1}$ for each outer iteration of (9), only a single (inner) iteration of (21) is sufficient for small problem sizes (e.g. $n_p = 5$).

To summarize, an iteration of the gradient projection method of (9) involves the following steps. Once an unconstrained step has been taken according to (13), the impulse strengths are transformed to modified switching instants according to (15). Then, the vector of modified switching instants $\boldsymbol{t}_p$ is projected onto the monotone cone with (19), which is then projected onto the box with (18). The modified switching instants are then transformed back to impulse strengths according to

$$\lambda_{p,i} = (t_{p,i} - t^*_{p,i})\Delta u_{p,i}$$

for the next iteration of the (outer) gradient projection method.

IV. Implementation

The entire control algorithm is implemented in very high speed integrated circuit hardware description language (VHDL). Although details regarding VHDL are omitted for the sake of brevity, the interested reader is referred to the excellent textbook [15] on how to use VHDL effectively.

A. Design Choices

The entire design of the control algorithm uses a single clock, which is a recommended design practice (see [15, Section 19.3.2]). It is decided that the design is clocked using a $50\,\mathrm{MHz}$ clock. For a controller with a sampling interval of $T_s = 25\,\mu\mathrm{s}$, this results in 1250 clock cycles being available to execute the control algorithm. A low-cost Xilinx XC7Z020 Zynq-7000 FPGA is used, which has 220 DSPE41 slices; each DSP slice contains a single 25×18 bit multiplier.[3] Next, some of the design choices of the control algorithm are discussed.

First, fixed-point arithmetic is used. When compared to floating-point arithmetic, fixed-point arithmetic requires less resources and results in faster computations. Furthermore, there is little reason to use floating-point arithmetic when considering the fact that model inaccuracies and quantization errors (from analogue-to-digital converters) are present in a practical system. Floating-point arithmetic may only be required if certain aspects of a control algorithm require high numerical precision for stability purposes. Thankfully, the gradient method of Section III can tolerate rounding errors [16], and numerical stability is achieved even with a fixed-point arithmetic.

Second, all variables are limited to a word length of 18 bits to minimize the usage of DSP slices. Using word lengths that exceed the capabilities of a single DSP slice require additional DSP slices to be invoked. Furthermore, 18 bits yielded sufficient numerical accuracy.

Third, special attention is given on how the DSP slices are utilized in the design. It is possible to multiply and use the answer within a *single* clock cycle. Doing so, however, will severely limit the clock speed the DSP slices can be operated at—and consequently the clock speed of the entire design. In order to operate a DSP slice at a high clock speed, a technique known as *pipelining* should be used; the result of a DSP slice is passed through a so-called *pipeline register* before it is used. For an illustrative example, refer to Fig. 4, where the multiplication $y = x_1x_2x_3x_4$ is shown. As seen, pipeline registers are placed at the output of each DSP slice.[4] Note that two clock cycles are required to *ready* the pipeline registers before the result y is available.

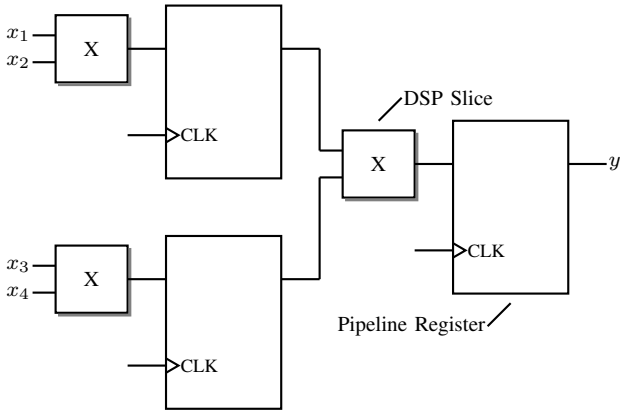


Fig. 4: Example of pipelining.

Fourth, the matrix exponentials are calculated offline, at a time resolution of $1\,\mu\mathrm{s}$ and over an interval of $-1\,\mathrm{ms}$ to $3\,\mathrm{ms}$ (for 18-bit words, this results in $158\,\mathrm{kB}$ of data), and stored in lookup tables. Furthermore, the steady-state trajectories are also calculated offline, at a time resolution of $25\,\mu\mathrm{s}$ and over one fundamental period (for 18-bit words, this results in $10\,\mathrm{kB}$ of data per steady-state trajectory), and stored in lookup tables.

Finally, recall that the problem size (the dimension of the decision variable $\boldsymbol{\Lambda}$) is time-varying and given as $n_{\mathrm{sw}} = n_a + n_b + n_c$ (the total number of switching transitions that fall within the prediction horizon T_p). The design on the FPGA must be of a fixed size, since the hardware required for the design has to be invoked during synthesis; once the FPGA is programmed, additional hardware cannot be invoked. The (fixed-size) implementation assumes a maximum of five switching transitions per phase at any given moment, leading to a maximum problem dimension of 15. For a pulse number of $d = 5$ and a prediction horizon of $T_p = 2\,\mathrm{ms}$, there are usually no more than three switching transitions per phase.

B. DSP Slice Usage and Execution Time

Out of the 220 DSP slices that are available, only 69 were required for the design: 50 for formulating the QP (that is, calculating the Hessian and the vector), 3 for determining the stepsize, and 16 for the gradient projection method.

To calculate the Hessian and the vector, 124 clock cycles are required. Calculating the stepsize (using 3 iterations of Newton's method) requires 24 clock cycles. Solving the QP

[3] Operations such as additions, subtractions, or bit shifting (which relates to multiplication or division by powers of two) typically require little resources to realize and can be implemented using (cheap) logic elements. On the other hand, multiplications are realized by the dedicated multipliers (DSP slices) within the FPGA.

[4] Additional pipeline registers may also be used, which may result in even higher clock speeds. However, a single pipeline register at the output enabled a DSP slice to be operated at $50\,\mathrm{MHz}$.

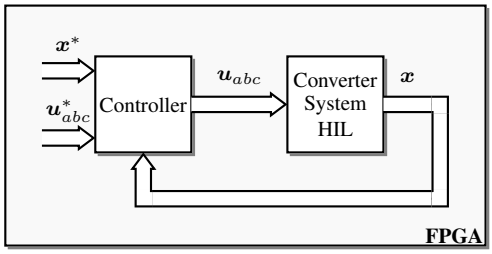


Fig. 5: Setup of the controller and HIL on the FPGA, where $\boldsymbol{x}^*$ denotes the steady-state (reference) trajectories, and $\boldsymbol{u}^*_{abc}$ and $\boldsymbol{u}_{abc}$ are the nominal and modified optimized pulse patterns, respectively.

with the gradient projection method (using 35 iterations) requires 921 clock cycles. In total, 1072 clock cycles are required to execute the entire control algorithm, translating to $21.4\,\mu$s, which is within the sampling interval of $T_s = 25\,\mu$s.

V. Results

To validate the FPGA implementation, a HIL simulation is considered [17]. The HIL simulation is realized using a discrete-time state-space representation (with a sampling interval of $500\,$ns) of the converter system, which is implemented using the remaining resources on the FPGA. Fixed-point variables with a word length of 32 bits are used to accurately simulate the system. Note that the HIL simulation assumes an idealized setup: effects such as measurement errors, parameter variation, deadtime, delays, dc-link voltage ripple, fluctuations of the neutral-point potential, and so on, are not present. A diagram of the HIL setup on the FPGA is shown in Fig. 5.

A medium-voltage grid-connected converter system is used; the parameters can be found in Appendix C. The system possesses two resonances at $262\,$Hz and $491\,$Hz. The sampling interval and prediction horizon of the controller are set to $T_s = 25\,\mu$s and $T_p = 2\,$ms, respectively. All state variables are penalized equally with $\boldsymbol{Q} = \mathbf{I}_6$, and all switching instant modifications are also penalized equally with $\boldsymbol{R} = 2\mathbf{I}_{n_{\text{sw}}}$. The pulse number $d = 5$ is used, which results in a device switching frequency of $f_{\text{sw}} = df_1 = 250\,$Hz (which is a typical switching frequency of a medium-voltage converter system) for a $f_1 = 50\,$Hz grid. The OPPs are calculated such that the harmonic distortions of the grid currents are minimized, considering the filter transfer function from the switching function to the grid current harmonics. The resulting (primary) switching angles are shown in Fig. 6.

All waveforms are normalized using a per-unit (pu) system, whose base values can be found in Appendix C.

A. Steady-State Performance

In Fig. 7, the steady-state waveforms of the grid current and switch positions are shown. The converter is operating at rated power with unity power factor (that is, $P = 1\,$pu and $Q = 0\,$pu). During steady-state conditions, the controller makes no adjustments to the nominal pulse pattern. Thus, the steady-state performance is entirely dictated by the (offline-calculated) nominal pulse pattern, which achieves an optimal ratio of harmonic distortions per switching frequency. The total demand distortion of the grid current is $I_{g,\text{TDD}} = 1.57\,\%$.

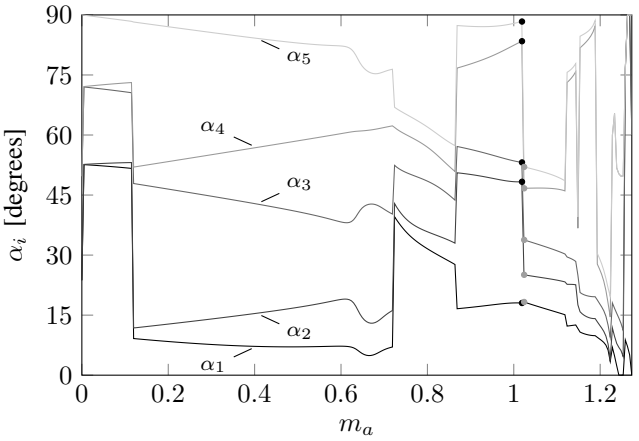


Fig. 6: The optimal switching angles $\boldsymbol{\alpha}$ for pulse number $d = 5$.

B. Quasi-Steady-State Performance

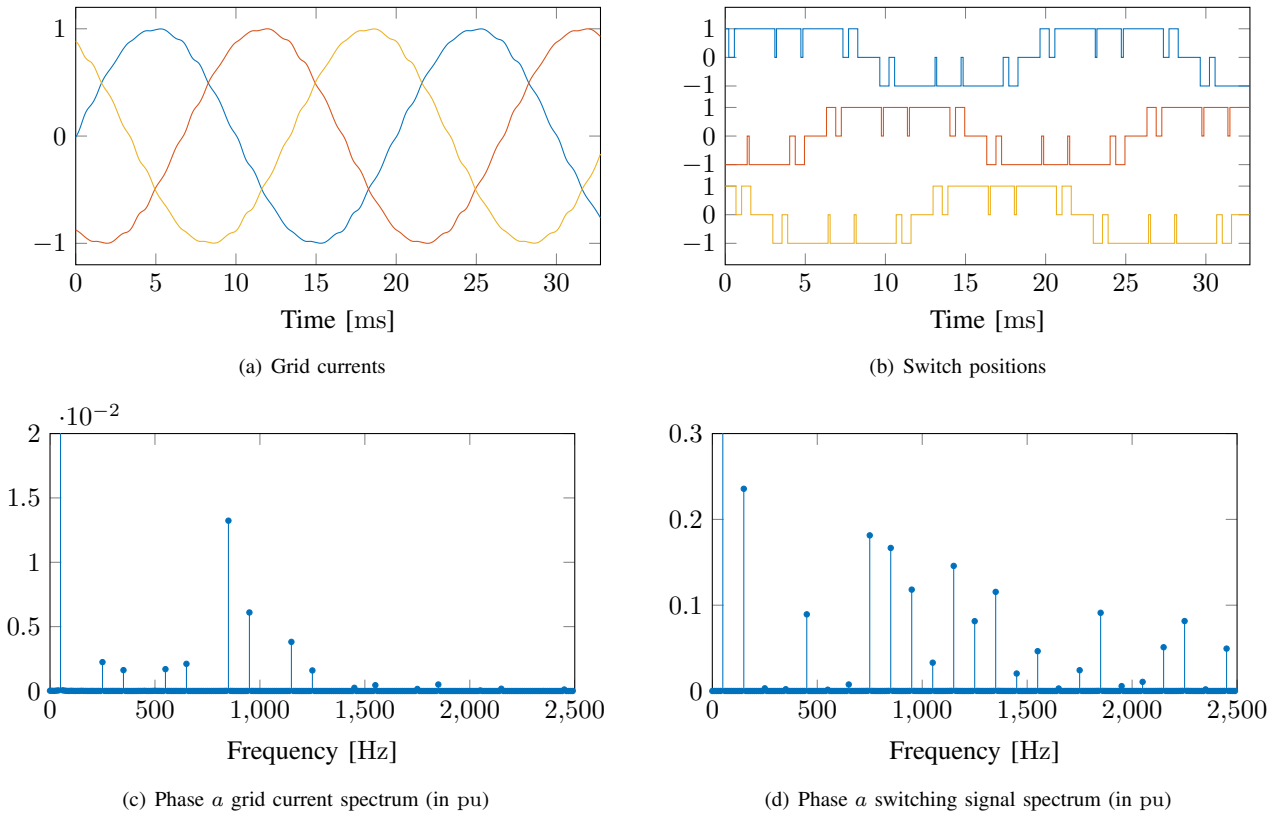
A practical converter system rarely operates in pure steady-state conditions due to unmeasured and unmodelled disturbances, leading to the modulation index to have small perturbations. In the case that the operating point of a converter system is at a discontinuity of the switching angles, it is paramount that the controller is able to react fast when the operating point is moved past such discontinuities.

Consider Fig. 8. Initially, a pulse pattern with a modulation index of $m_a = 1.019$ is applied to the converter system. This modulation index coincides with the switching angles that are marked with the black dots in Fig. 6. At $15\,$ms, the modulation index changes slightly to $m_a = 1.024$, which causes a discontinuity in the switching angles (marked with the gray dots). As seen in Fig. 8, the controller quickly rejects any excursions in the converter states and promptly regulates the state variables onto their (new) steady-state (reference) trajectories; the error never exceeds $1.25\,\%$ and is reduced to less than $1\,\%$ within $0.72\,$ms (or 29 controller sampling intervals). It can be observed in Fig. 8(d) that only very small adjustments are made to the pulse patterns. Without closed-loop control, multiple fundamental periods (roughly eight) are required for the excursions to diminish due to the very small ohmic resistances in the system.

C. Performance During a Reference Step

In Fig. 9, the converter is initially operating at rated power and at unity power factor. The (real) power reference is stepped to $P^* = 0\,$pu at $5\,$ms, and then back to $P^* = 1\,$pu at $17\,$ms. The controller quickly regulates all state variables to their respective references. During the transients, it can be observed from Fig. 9(d) that the nominal pulse pattern is modified significantly; for example, note that from $17\,$ms to $22\,$ms some of the switching transitions are removed. Once the converter enters the steady state, the (unmodified) nominal pulse pattern is applied to the converter system and the superior harmonic performance of the OPPs is achieved.

Although there are some oscillations present during the transients, it should be noted that the converter system is underdamped and, due to the low switching frequency, there



(a) Grid currents

(b) Switch positions

(c) Phase a grid current spectrum (in pu)

(d) Phase a switching signal spectrum (in pu)

Fig. 7: Steady-state operation.

are only a few switching transitions available that can be used to achieve closed-loop control. It is likely that the response time can be reduced by inserting additional switching transitions; this is known as *pulse insertion* and the MP^{3}C algorithm utilizes this technique, see [4, Section 12.6] and [18].

D. Performance During a Grid Fault

The converter system is initially operating at rated power and unity power factor in Fig. 10. At $5\,\mathrm{ms}$, a (symmetrical) three-phase-to-ground short-circuit fault occurs and the grid phase voltage drops to $0.05\,\mathrm{pu}$. In order to feed the short circuit, the grid current setpoint is increased to $1.2\,\mathrm{pu}$. Note that the new steady-state trajectory considers the now-reduced grid voltage. As seen, the controller quickly rejects the fault and feeds the short circuit. This is typically required in small or islanded grids—with significant power electronic sources and loads—to ensure selectivity of protective devices during short circuits. In particular, the circuit breaker closest to the fault is meant to isolate the faulty equipment, minimizing the interruption to other loads. By feeding a significant current into the short circuit for up to a few $100\,\mathrm{ms}$, the converter system supports selectivity of the protective devices.

VI. Conclusions

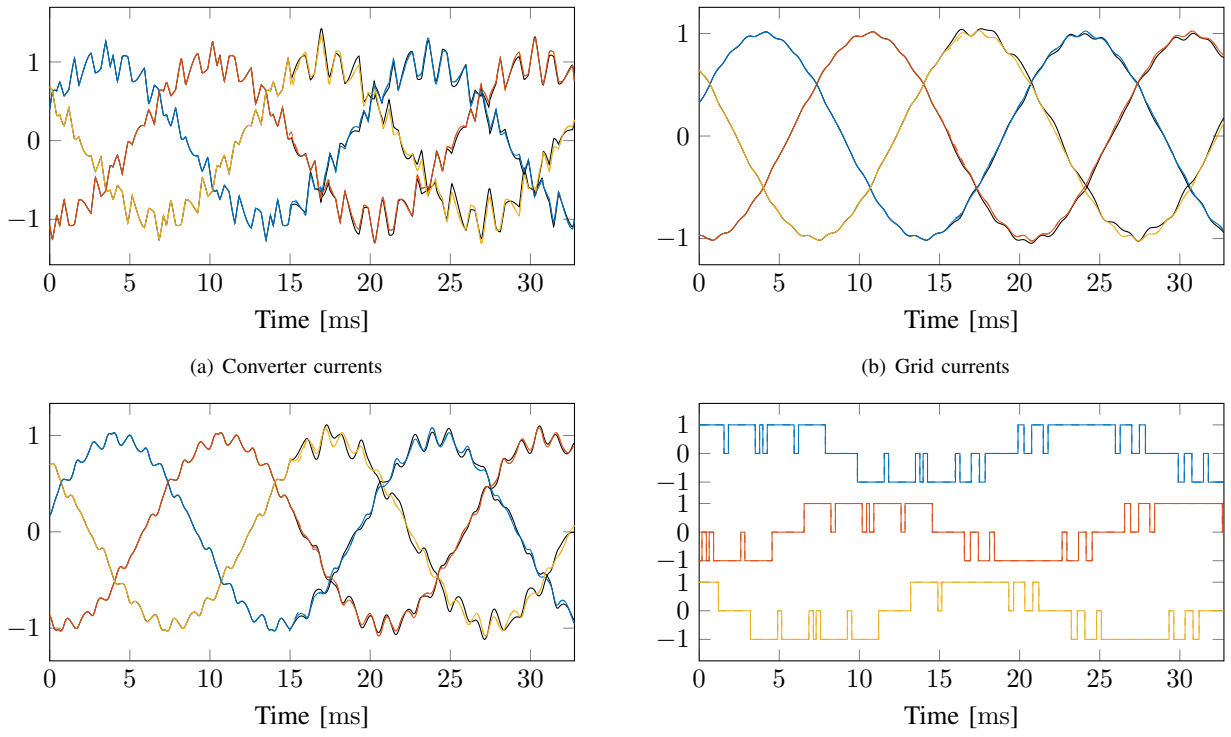
This paper focuses on the implementation of the QP underlying the proposed controller from [5]. The main challenges are the (relatively) high dimension of 15 of the decision variable, the very short sampling interval of $25\,\mu\mathrm{s}$, and the time-varying nature of the Hessian matrix. To address these challenges, recommendations were provided to reduce the computational burden of the QP with the aim to enable an efficient implementation on an FPGA.

In particular, the number of calculations required during the QP formulation can be reduced by exploiting the algebraic structure of the control problem and the characteristics of the load. It was further shown that the Lipschitz constant can be overestimated using the infinity-norm of the Hessian, and how the stepsize of the gradient method can be computed using Newton's method. Finally, it was explained how the gradient projection method can be efficiently implemented. Using these recommendations, an efficient implementation of the standard controller was achieved on a low-cost FPGA, as only 69 DSP slices were required. The control algorithm required $21.4\,\mu\mathrm{s}$ to execute.

The performance of the controller was evaluated by using (idealized) HIL simulations. Results showed a controller with a fast dynamic response during transients, such as reference steps and faults. During steady-state operation, the controller applies the nominal pulse patterns, thus achieving a superior harmonic performance.

Appendix A
Reducing the Computational Burden of the QP Formulation

In this appendix the structure of the Hessian $\boldsymbol{H}$ and vector $\boldsymbol{c}$ are analyzed and exploited in order to reduce the



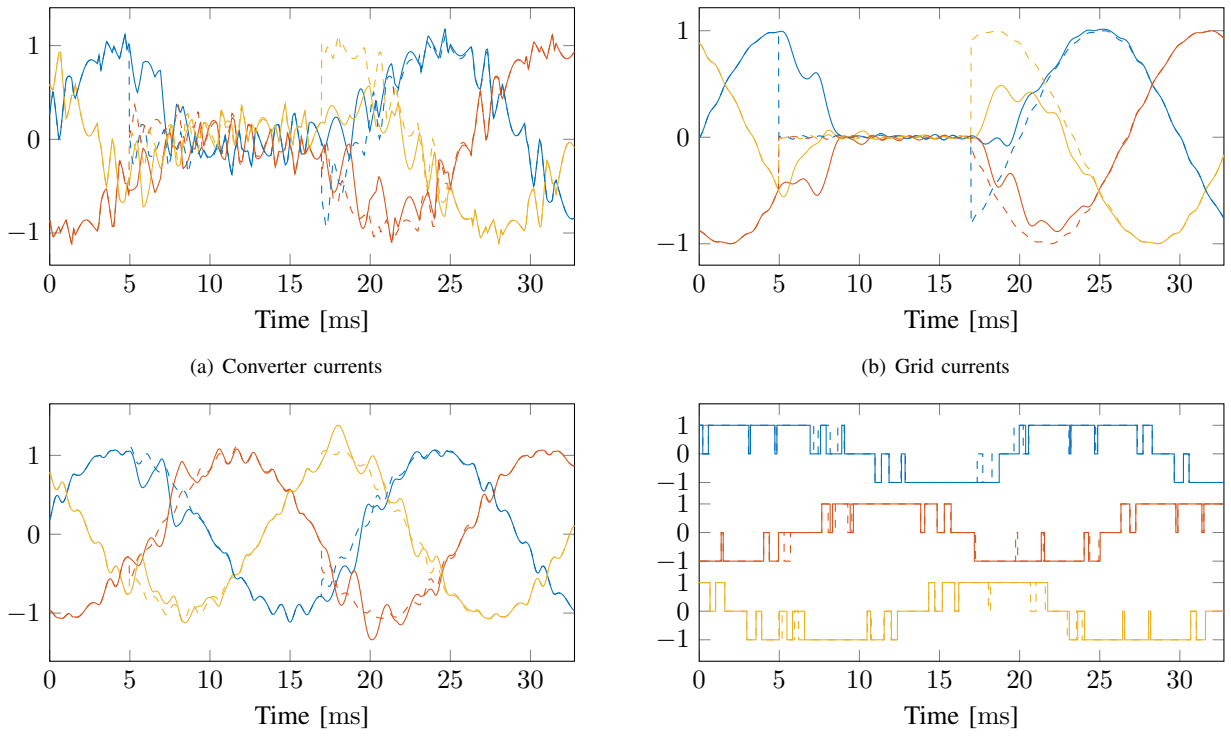
(a) Converter currents

(b) Grid currents

(c) Capacitor voltages

(d) Switch positions

Fig. 8: Quasi-steady-state operation. The open-loop responses are shown by the black lines. The steady-state (reference) trajectories are indicated by dashed lines.



(a) Converter currents

(b) Grid currents

(c) Capacitor voltages

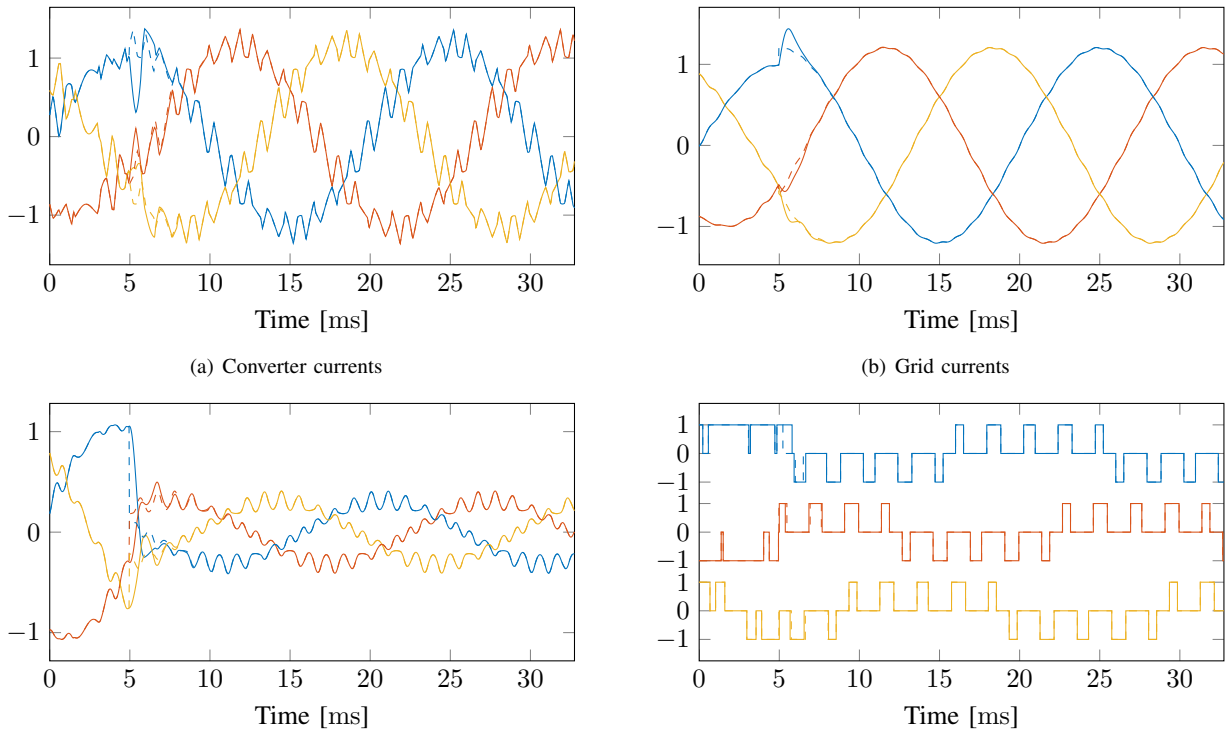
(d) Switch positions

Fig. 9: Reference steps. The steady-state (reference) trajectories are indicated by dashed lines.

computational burden with the aim of enabling an efficient implementation on an FPGA.

A. *Exploiting the Problem Structure*

In a first step, to reduce the computational burden, several aspects of the problem can be exploited. First, since $t_{ij}^* =$



(a) Converter currents

(b) Grid currents

(c) Capacitor voltages

(d) Switch positions

Fig. 10: Grid fault. The steady-state (reference) trajectories are indicated by dashed lines.

$\max\{t^*_{p_1,i}, t^*_{p_2,j}\}$, it can be observed that either

$$\boldsymbol{e}^{\boldsymbol{F}^{\mathrm{T}}(t^*_{ij}-t^*_{p_1,i})} \quad \text{or} \quad \boldsymbol{e}^{\boldsymbol{F}(t^*_{ij}-t^*_{p_2,j})}$$

of (6) will be the identity matrix. This means that one of the matrix-vector products

$$\boldsymbol{G}_{p_1}^{\mathrm{T}}\boldsymbol{e}^{\boldsymbol{F}^{\mathrm{T}}(t^*_{p,ij}-t^*_{p_1,i})} \quad \text{or} \quad \boldsymbol{e}^{\boldsymbol{F}(t^*_{ij}-t^*_{p_2,j})}\boldsymbol{G}_{p_2},$$

where each product may require up to 36 multiplications, is not required to be calculated. For the elements on the diagonal of $\boldsymbol{V}$, $i' = j'$ holds. This implies that $p = p_1 = p_2$ and $t^*_{p,i} = t^*_{p_1,i} = t^*_{p_2,j}$. This leads to both

$$\boldsymbol{e}^{\boldsymbol{F}^{\mathrm{T}}(t^*_{ij}-t^*_{p_1,i})} \quad \text{and} \quad \boldsymbol{e}^{\boldsymbol{F}(t^*_{ij}-t^*_{p_2,j})}$$

being identity matrices, and the diagonal components of $\boldsymbol{V}$ have the form

$$\boldsymbol{V}_{(i',j')} = \boldsymbol{G}_p^{\mathrm{T}}\boldsymbol{\Xi}(T_p - t^*_{p,i})\boldsymbol{G}_p. \tag{22}$$

Finally, note that the term

$$\boldsymbol{\Xi}(T_p - t^*_{p,i})\boldsymbol{G}_p$$

of (22) is present in (7). Thus, half of the terms of $\boldsymbol{c}_{i'}$ have already been calculated when the diagonal components of $\boldsymbol{V}$ are calculated.

B. Exploiting the Load Characteristics

Next, characteristics of the load are exploited in order to reduce the computational burden even further. Due to the α and β components of the system matrix $\boldsymbol{F}$ being decoupled, half of the components of the matrix exponential $\boldsymbol{e}^{\boldsymbol{F}t}$ are zero; it is of the structure

$$\boldsymbol{e}^{\boldsymbol{F}t} = \begin{bmatrix} \mathrm{x} & 0 & \mathrm{x} & 0 & \mathrm{x} & 0 \\ 0 & \mathrm{x} & 0 & \mathrm{x} & 0 & \mathrm{x} \\ \mathrm{x} & 0 & \mathrm{x} & 0 & \mathrm{x} & 0 \\ 0 & \mathrm{x} & 0 & \mathrm{x} & 0 & \mathrm{x} \\ \mathrm{x} & 0 & \mathrm{x} & 0 & \mathrm{x} & 0 \\ 0 & \mathrm{x} & 0 & \mathrm{x} & 0 & \mathrm{x} \end{bmatrix},$$

where x denotes a nonzero element. Thus, the term

$$\tilde{\boldsymbol{x}}_0^{\mathrm{T}}\boldsymbol{e}^{\boldsymbol{F}^{\mathrm{T}}t^*_{p,i}}$$

of (7) only requires 18 multiplications (instead of 36). Note that for any given phase $p \in \{a, b, c\}$ only the top two elements of $\boldsymbol{G}_p$ are nonzero, see (8). This results in matrix-vector multiplications in (6) of the forms

$$\boldsymbol{e}^{\boldsymbol{F}t}\boldsymbol{G}_p \quad \text{and} \quad \boldsymbol{\Xi}(\Delta T)\boldsymbol{G}_p$$

only requiring 6 multiplications (instead of 36).

Appendix B
Matrix Form of Constraints

The matrix $\boldsymbol{A}_p$ and vector $\boldsymbol{b}_p$ are defined as

$$\boldsymbol{A}_p = \begin{bmatrix} \frac{1}{\Delta u_{p,1}} & 0 & 0 & \cdots & 0 & 0 \\ -\frac{1}{\Delta u_{p,1}} & \frac{1}{\Delta u_{p,2}} & 0 & \cdots & 0 & 0 \\ 0 & -\frac{1}{\Delta u_{p,2}} & \frac{1}{\Delta u_{p,3}} & \cdots & 0 & 0 \\ \vdots & & & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -\frac{1}{\Delta u_{p,n_p-1}} & \frac{1}{\Delta u_{p,n_p}} \\ 0 & 0 & 0 & \cdots & 0 & \frac{1}{\Delta u_{p,n_p}} \end{bmatrix}$$

TABLE I: System parameters of the medium-voltage case study.

Parameter	Symbol	Value	Per unit
Converter			
Dc-link voltage	V_d	4840 V	1.8818 pu
Rated apparent power	S_R	9 MVA	1 pu
Half dc-link capacitance	C_d	9.9 mF	3.4290 pu
Filter inductance	L	350 µH	0.0997 pu
Filter capacitance	C	420 µF	0.1455 pu
Filter inductor ESR	R	0.3 mΩ	0.000 27 pu
Filter capacitor ESR	R_C	4 mΩ	0.0036 pu
Transformer inductance	L_t	526.41 µH	0.15 pu
Transformer resistance	R_t	16.54 mΩ	0.015 pu
Grid			
Grid-side inductance	L_g	349.19 µH	0.0995 pu
Grid-side inductor ESR	R_g	10.97 mΩ	0.010 pu
Fundamental frequency	f_1	50 Hz	1 pu
Grid voltage (line-to-line)	V_g	3150 V (rms)	1.2247 pu
Rated phase current	I_R	1649.6 A (rms)	0.7071 pu

and

$$\boldsymbol{b}_p = \begin{bmatrix} t^*_{p,1} & t^*_{p,2} - t^*_{p,1} & \cdots & T_p - t^*_{p,n_p} \end{bmatrix}^{\mathrm{T}},$$

respectively.

Appendix C
Parameters

The parameters of a 9 MVA grid-connected NPC converter with an LC filter are shown in Table I. Integrated-gate-commutated thyristors (IGCTs) are depicted in Fig. 2, since they are the preferred choice of semiconductors at these power levels. IGCTs offer very low on-state losses, an output voltage of 3.15 kV is achieved without series-connection of devices, and IGCTs allow high thermal cycling due to their press-pack housing. A per-unit system is established with the following base values: $V_B = \sqrt{\frac{2}{3}} V_g$, $I_B = \sqrt{2} I_R$, and $\omega_B = \omega_1$.

References

[1] G. S. Buja and G. B. Indri, "Optimal pulsewidth modulation for feeding AC motors," *IEEE Transactions on Industry Applications*, vol. IA-13, pp. 38–44, Jan. 1977.

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

[3] J. Scoltock, T. Geyer, and U. K. Madawala, "A comparison of model predictive control schemes for MV induction motor drives," *IEEE Transactions on Industrial Informatics*, vol. 9, pp. 909–919, May 2013.

[4] T. Geyer, *Model predictive control of high power converters and industrial drives*. Chichester, UK: John Wiley & Sons, Ltd, Nov. 2016.

[5] M. Dorfling, H. d. T. Mouton, and T. Geyer, "Generalized Model Predictive Pulse Pattern Control Based on Small-Signal Modelling—Part 1: Algorithm," submitted to the Transactions on Power Electronics.

[6] T. Geyer, N. Oikonomou, G. Papafotiou, and F. D. Kieferndorf, "Model predictive pulse pattern control," *IEEE Transactions on Industry Applications*, vol. 48, pp. 663–676, Mar. 2012.

[7] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer Series in Operations Research and Financial Engineering, New York: Springer, 2nd ed ed., 2006.

[8] D. P. Bertsekas, *Nonlinear programming*. Belmont, Massachusetts: Athena Scientific, 3rd ed., 2016.

[9] A. Beck, *Introduction to nonlinear optimization: theory, algorithms, and applications with Matlab*. Philadelphia, PA: Society for Industrial and Applied Mathematics, Oct. 2014.

[10] G. H. Golub and C. F. Van Loan, *Matrix computations*. Johns Hopkins studies in the mathematical sciences, Baltimore: The Johns Hopkins University Press, 4th ed., 2013.

[11] M. Flynn, "On division by functional iteration," *IEEE Transactions on Computers*, vol. C-19, pp. 702–706, Aug. 1970.

[12] R. L. Burden, J. D. Faires, and A. M. Burden, *Numerical analysis*. Boston, MA: Cengage Learning, 10th ed., 2016.

[13] S. Richter, T. Geyer, and M. Morari, "Resource-efficient gradient methods for model predictive pulse pattern control on an FPGA," *IEEE Transactions on Control Systems Technology*, vol. 25, pp. 828–841, May 2017.

[14] A. Németh and S. Németh, "How to project onto an isotone projection cone," *Linear Algebra and its Applications*, vol. 433, pp. 41–51, July 2010.

[15] R. Jasinski, *Effective coding with VHDL: principles and best practice*. Cambridge, Massachusetts: The MIT Press, 2016.

[16] O. Devolder, F. Glineur, and Y. Nesterov, "First-order methods of smooth convex optimization with inexact oracle," *Mathematical Programming*, vol. 146, pp. 37–75, Aug. 2014.

[17] N. Roshandel Tavana and V. Dinavahi, "A general framework for FPGA-based real-time emulation of electrical machines for HIL applications," *IEEE Transactions on Industrial Electronics*, vol. 62, pp. 2041–2053, Apr. 2015.

[18] T. Geyer and N. Oikonomou, "Model predictive pulse pattern control with very fast transient responses," in *2014 IEEE Energy Conversion Congress and Exposition (ECCE)*, (Pittsburgh, PA, USA), pp. 5518–5524, IEEE, Sept. 2014.