

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

Tinus Dorfling <sup>ib</sup>, Hendrik du Toit Mouton <sup>ib</sup>, *Member, IEEE*, and Tobias Geyer <sup>ib</sup>, *Fellow, IEEE*

**Abstract**—A newly proposed model predictive pulse pattern controller, which is applicable to higher-order converter systems, is implemented on a low-cost field-programmable gate array. It is shown that 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**—Field-programmable gate arrays (FPGAs), model predictive control, optimized pulse patterns (OPPs).

## 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 and 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 particularly 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 article [5], a generalization of the model predictive pulse pattern controller [6] is 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  $\mathbf{x}$  of a converter system along its optimal steady-state trajectory  $\mathbf{x}^*$  by modifying the switching instants of the nominal pulse pattern

Manuscript received May 26, 2021; revised November 14, 2021 and February 10, 2022; accepted April 8, 2022. Date of publication April 22, 2022; date of current version May 23, 2022. Recommended for publication by Associate Editor M. A. Perez. (*Corresponding author: Tinus Dorfling.*)

Tinus Dorfling and Hendrik du Toit Mouton are with the Department of Electrical and Electronic Engineer, Stellenbosch University, Stellenbosch 7599, South Africa (e-mail: martinusdorfling@protonmail.com; dtmouton@sun.ac.za).

Tobias Geyer is with the ABB System Drives, 5300 Turgi, Switzerland (e-mail: t.geyer@ieee.org).

Color versions of one or more figures in this article are available at <https://doi.org/10.1109/TPEL.2022.3169119>.

Digital Object Identifier 10.1109/TPEL.2022.3169119

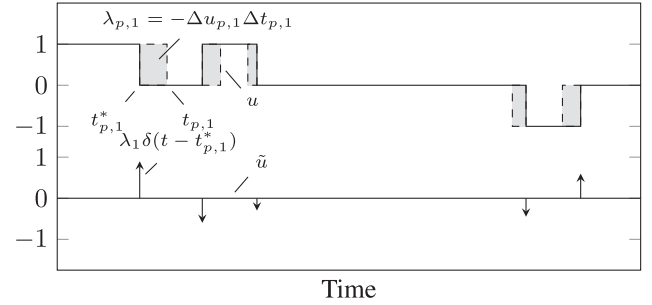


Fig. 1. Impulses represent areas that are added or removed from a pulse pattern. One phase of a three-level pulse pattern is shown.

$\mathbf{u}_{abc}^*$ , resulting in a modified pulse pattern  $\mathbf{u}_{abc}$ . It was shown that the strengths of impulses can be used to represent the areas, which are added or removed to a pulse pattern (in other words, the modifications) as

$$\lambda_{p,i} = -\Delta t_{p,i} \Delta u_{p,i} \quad (1)$$

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  $\Delta u_{p,i} = u_{p,i} - u_{p,i-1}$  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  $i$ th 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]

$$\mathbf{\Lambda}_{\text{opt}} = \arg \min_{\mathbf{\Lambda}} \frac{1}{2} \mathbf{\Lambda}^T \mathbf{H} \mathbf{\Lambda} + \mathbf{c}^T \mathbf{\Lambda} \quad (2a)$$

$$\text{subject to } \mathbf{A} \mathbf{\Lambda} \leq \mathbf{b}. \quad (2b)$$

The decision vector  $\mathbf{\Lambda} \in \mathbb{R}^{n_{\text{sw}}}$ , 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

$$\mathbf{\Lambda} = [\lambda_{a,1} \cdots \lambda_{a,n_a} \lambda_{b,1} \cdots \lambda_{b,n_b} \lambda_{c,1} \cdots \lambda_{c,n_c}]^T. \quad (3)$$

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

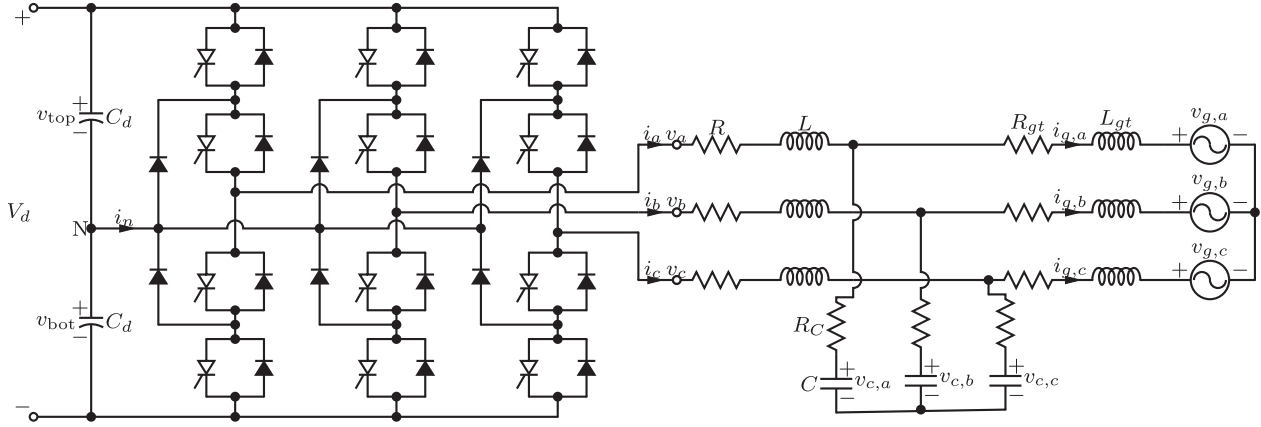


Fig. 2. Grid-connected NPC converter with an  $LC$  filter. The resistances and inductances of the grid and transformer are lumped together as  $R_{gt} = R_g + R_t$  and  $L_{gt} = L_g + L_t$ , respectively. Note that the resistors represent the equivalent series resistances of the filter inductor and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

modifications

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

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

$$t_{\text{opt},p,i} = t_{p,i}^* + \Delta t_{\text{opt},p,i}. \quad (5)$$

This article 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 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 article 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  $\mathbf{H} \in \mathbb{R}^{n_{\text{sw}} \times n_{\text{sw}}}$  and the vector  $\mathbf{c} \in \mathbb{R}^{n_{\text{sw}}}$  of the quadratic objective function. It is important to note that  $\mathbf{H}$  and  $\mathbf{c}$  have to be calculated at *every* sampling instant of the control algorithm, as they are not constant.

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

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

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

$$\Xi(\Delta T) = \int_0^{\Delta T} \mathbf{e}^{\mathbf{F}^T t} \mathbf{Q} \mathbf{e}^{\mathbf{F} t} dt.$$

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

$$\mathbf{c}_{i'}^T = \tilde{\mathbf{x}}_0^T \mathbf{e}^{\mathbf{F}^T t_{p_1,i}^*} \Xi(T_p - t_{p_1,i}^*) \mathbf{G}_{p_1}. \quad (7)$$

Recall that  $\mathbf{F} \in \mathbb{R}^{6 \times 6}$  and  $\mathbf{G} \in \mathbb{R}^{6 \times 3}$  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<sup>1</sup>

$$\mathbf{F} = \begin{bmatrix} -\frac{R+R_C}{L} \mathbf{I}_2 & \frac{R_C}{L} \mathbf{I}_2 & -\frac{1}{L} \mathbf{I}_2 \\ \frac{R_C}{L_{gt}} \mathbf{I}_2 & -\frac{R_{gt}+R_C}{L_{gt}} \mathbf{I}_2 & \frac{1}{L_{gt}} \mathbf{I}_2 \\ \frac{1}{C} \mathbf{I}_2 & -\frac{1}{C} \mathbf{I}_2 & \mathbf{0}_{2 \times 2} \end{bmatrix}$$

$$\mathbf{G} = \frac{V_d}{2L} \begin{bmatrix} \mathbf{I}_2 \\ \mathbf{0}_{2 \times 2} \\ \mathbf{0}_{2 \times 2} \end{bmatrix} \mathbf{K} \quad (8)$$

for a state vector, which is defined as

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

<sup>1</sup>Matrix  $\mathbf{K} = \frac{2}{3} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix}$  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  $2 \times 2$  zero matrix.

Furthermore,  $\mathbf{G}_a = \mathbf{G}[1 \ 0 \ 0]^T$ ,  $\mathbf{G}_b = \mathbf{G}[0 \ 1 \ 0]^T$ , and  $\mathbf{G}_c = \mathbf{G}[0 \ 0 \ 1]^T$  are the 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 given 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  $\mathbf{H}$  and the vector  $\mathbf{c}$  have been calculated, the QP of (2) can be solved using the gradient projection method. The  $k$ th iteration of the method, with step size  $s$  and gradient  $\nabla f(\mathbf{\Lambda}_k) = \mathbf{H}\mathbf{\Lambda}_k + \mathbf{c}$ , is

$$\mathbf{\Lambda}_{k+1} = \pi_{\mathcal{Z}}(\mathbf{\Lambda}_k - s(\mathbf{H}\mathbf{\Lambda}_k + \mathbf{c})) \quad (9)$$

where the feasible set is  $\mathcal{Z} = \{\mathbf{z} : \mathbf{A}\mathbf{z} \leq \mathbf{b}\}$ . The operator  $\pi_{\mathcal{Z}}(\cdot)$  projects the result onto the set  $\mathcal{Z}$ . In this article, a fixed step size of  $s = \frac{1}{L_c}$  is used for the gradient method, where  $L_c = \|\mathbf{H}\|_2$  is known as the tight Lipschitz constant. Here,  $\|\mathbf{H}\|_2$  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 step size  $s$  can be efficiently calculated; this is required at each sampling instant, since  $\mathbf{H}$  is time-varying. Once the step size  $s$  has been calculated, it is then shown that how the projection operation onto the set  $\mathcal{Z}$  can be efficiently realized.

#### A. Efficiently Determining the Step Size

In order to calculate the step size

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

two operations are required: 1) the Lipschitz constant  $L_c$  has to be calculated; and 2) calculating its reciprocal in order to determine the step size  $s$ .

1) *Efficiently Overestimating the Lipschitz Constant:* Consider the Lipschitz constant  $L_c$ , which requires the evaluation of the (induced) 2-norm of the Hessian  $\mathbf{H}$ . 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

$$\|\mathbf{H}\|_2 \leq \sqrt{\|\mathbf{H}\|_1 \|\mathbf{H}\|_\infty}.$$

And, since the Hessian  $\mathbf{H}$  is symmetric, the relation simplifies to

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

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

$$\|\mathbf{H}\|_\infty = \max_{1 \leq i \leq n_{sw}} \sum_{j=1}^{n_{sw}} |\mathbf{H}_{(i,j)}|, \quad (10)$$

which is the maximum of the absolute value of the matrix entries summed up in each row; no multiplications are required.

Denote with  $\bar{L}_c = \|\mathbf{H}\|_\infty$  the overestimated (tight) Lipschitz constant. An overestimated tight Lipschitz constant  $\bar{L}_c$  results in an underestimated step size  $s$ .

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

In a first step, the Lipschitz constant  $\bar{L}_c$  is normalized to the interval  $[0.5, 1]$  (this is a 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 2. 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,

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

can be approximated with  $z$  by finding the root of

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

by using the Newton's method, where the  $k$ th iteration has the form [11]

$$z_{k+1} = z_k + z_k(1 - Dz_k). \quad (11)$$

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

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 the 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, which is close to the solution  $\frac{1}{D}$ , significantly increases the convergence, since a small initial error (given by  $\epsilon_0 = 1 - Dz_0$ ) will rapidly diminish at subsequent iterations. To generate an initial solution  $z_0$ , consider the linear least-squares approximation problem

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

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

$$\begin{aligned} a_0 \int_{0.5}^1 1 dD + a_1 \int_{0.5}^1 D dD &= \int_{0.5}^1 \frac{1}{D} dD \\ a_0 \int_{0.5}^1 D dD + a_1 \int_{0.5}^1 D^2 dD &= \int_{0.5}^1 1 dD. \end{aligned}$$

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 the 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)

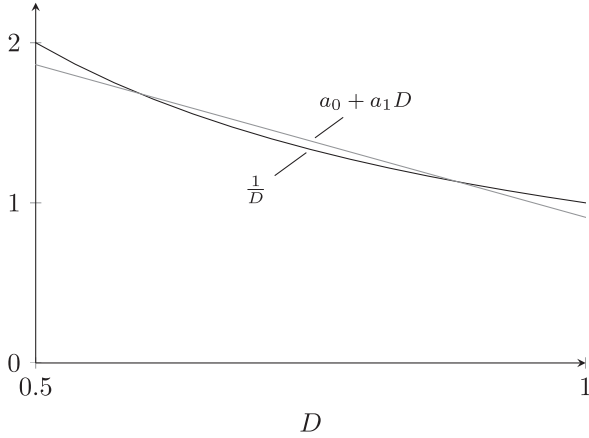


Fig. 3. Linear least square approximation of  $\frac{1}{D}$  in the region  $D \in [0.5, 1]$ . The coefficients are  $a_0 = 2.8162$  and  $a_1 = -1.9066$ .

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

$$\Lambda_k - \frac{1}{L_c} (\mathbf{H}\Lambda_k + \mathbf{c}) \quad (13)$$

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  $\mathcal{Z} = \{z : \mathbf{A}z \leq \mathbf{b}\}$  translate to

$$0 \leq t_{p,1} \leq t_{p,2} \leq \dots \leq t_{p,n_p} \leq T_p \quad (14)$$

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

$$t_{p,i} = t_{p,i}^* + \frac{\lambda_{p,i}}{\Delta u_{p,i}} \quad (15)$$

(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.

In the following section, it is shown how to efficiently project onto the set of (14). Note that 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  $\mathbf{t}_p \in \mathbb{R}^{n_p}$  the vector of modified switching instants, which 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

$$\bar{\mathbb{K}} = \{z : z \leq z_1 \leq z_2 \leq \dots \leq z_{n_z} \leq \bar{z}\} \quad (16)$$

where  $z$  and  $\bar{z}$  are the lower and upper bounds, respectively.<sup>2</sup> 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

$$\mathbb{K} = \{z : z_1 \leq z_2 \leq \dots \leq z_{n_z}\}$$

$$\mathbb{B} = \{z : z \leq z_i \leq \bar{z} \text{ for } i = 1, 2, \dots, n_z\}$$

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

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

The second projection, onto the box  $\mathbb{B}$ , is extremely simple to realize; the projection of the  $i$ th component of  $\xi$  is simply

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

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

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

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

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 article. Fortunately, Richter *et al.* [13] proposed a computationally efficient approximation that requires no (pseudo)inverses of matrices; due to this advantage, the approach in [13] is thus adopted in this article. As shown in [13, Equation (25a)], the approximated projection follows as

$$\pi_{\mathbb{K}}(\mathbf{t}_p) = \mathbf{t}_p - \mathbf{C}^T \eta_{\text{opt}} \quad (19)$$

with

$$\eta_{\text{opt}} = \arg \min_{\eta \geq 0} \frac{1}{2} \eta^T \mathbf{C} \mathbf{C}^T \eta - (\mathbf{C} \mathbf{y})^T \eta \quad (20)$$

<sup>2</sup>In the case of (14), the lower and upper bounds are  $z = 0$  and  $\bar{z} = T_p$ , respectively.

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, which 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  $\mu_d$  eigenvalues of the (dual) Hessian  $\mathbf{C}\mathbf{C}^T \in \mathbb{R}^{(n_p-1) \times (n_p-1)}$ , which is a second-order difference matrix,

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

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  $k$ th (inner) iteration is

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

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 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  $\mathbf{t}_p$  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 [15, Section 19.3.2]. It is decided that the design is clocked using a 50 MHz clock. For a controller with a sampling interval of  $T_s = 25 \mu\text{s}$ , this results in 1250 clock cycles being available to execute the control algorithm. A low-cost Xilinx XC7Z020 Zynq-7000 FPGA is

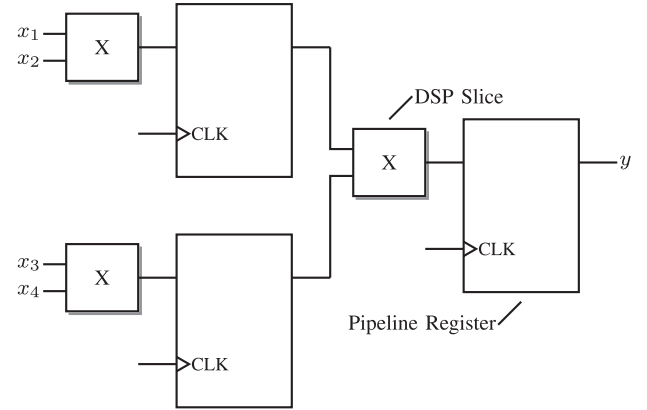


Fig. 4. Example of pipelining.

used, which has 220 DSPE41 slices; each digital signal processor (DSP) slice contains a single  $25 \times 18$  bit multiplier.<sup>3</sup> In the following, some of the design choices of the control algorithm are discussed.

- 1) First, fixed-point arithmetic is used. When compared to floating-point arithmetic, fixed-point arithmetic requires fewer 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.
- 2) 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.
- 3) 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  $y = x_1 x_2 x_3 x_4$  is shown. As seen, pipeline registers are placed at the output of each DSP slice.<sup>4</sup> Note that two clock cycles are

<sup>3</sup>Operations, such as additions, subtractions, or bit shifting, (which relates to multiplication or division by powers of 2) 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.

<sup>4</sup>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 MHz.

required to *ready* the pipeline registers before the result  $y$  is available.

- 4) Fourth, the matrix exponentials are calculated offline, at a time resolution of  $1 \mu\text{s}$  and over an interval of  $-1$  to  $3 \text{ ms}$  (for 18-bit words, this results in 158 kB of data), and stored in lookup tables. Furthermore, the steady-state trajectories are also calculated offline, at a time resolution of  $25 \mu\text{s}$  and over one fundamental period (for 18-bit words, this results in 10 kB of data per steady-state trajectory), and stored in lookup tables.
- 5) Finally, recall that the problem size (the dimension of the decision variable  $\Lambda$ ) is time-varying and given as  $n_{\text{sw}} = n_a + n_b + n_c$  (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  $T_p = 2 \text{ ms}$ , there are usually no more than three switching transitions per phase.

### B. DSP Slice Usage and Execution Time

The 220 DSP slices that are available, out of which only 69 were required for the design: 50 for formulating the QP (that is, calculating the Hessian and the vector), three for determining the step size, and 16 for the gradient projection method.

To calculate the Hessian and the vector, 124 clock cycles are required. Calculating the step size (using three iterations of the Newton's method) requires 24 clock cycles. Solving the QP 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  $21.4 \mu\text{s}$ , which is within the sampling interval of  $T_s = 25 \mu\text{s}$ .

## V. RESULTS

To validate the FPGA implementation, an HIL simulation is considered [17]. The HIL simulation is realized using a discrete-time state-space representation (with a sampling interval of  $500 \text{ ns}$ ) 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 and 491 Hz. The sampling interval and prediction horizon of the controller are set to  $T_s = 25 \mu\text{s}$  and  $T_p = 2 \text{ ms}$ , respectively. All state variables are penalized equally with  $\mathbf{Q} = \mathbf{I}_6$ , and all switching instant modifications are also penalized equally with  $\mathbf{R} = 2\mathbf{I}_{n_{\text{sw}}}$ . The pulse number

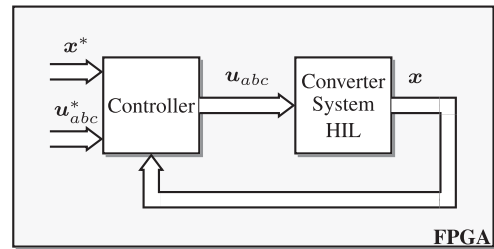


Fig. 5. Setup of the controller and HIL on the FPGA, where  $\mathbf{x}^*$  denotes the steady-state (reference) trajectories, and  $\mathbf{u}_{abc}^*$  and  $\mathbf{u}_{abc}$  are the nominal and modified OPPs, respectively.

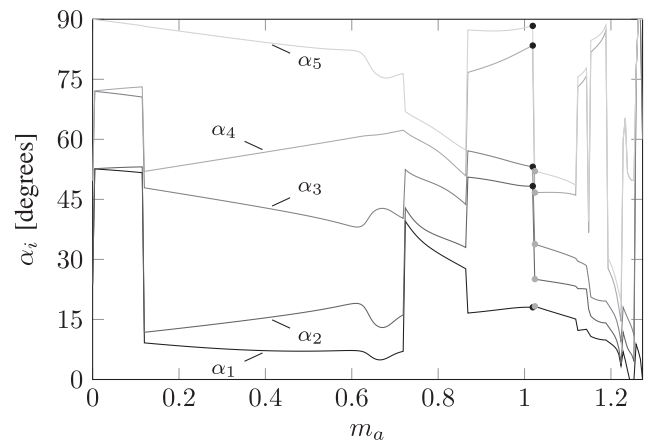


Fig. 6. Optimal switching angles  $\alpha$  for pulse number  $d = 5$ .

$d = 5$  is used, which results in a device switching frequency of  $f_{\text{sw}} = df_1 = 250 \text{ Hz}$  (which is a typical switching frequency of a medium-voltage converter system) for a  $f_1 = 50 \text{ Hz}$  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,  $P = 1$  and  $Q = 0 \text{ pu}$ ). 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  $I_{g,\text{TDD}} = 1.57 \%$ .

### B. Quasisteady-State Performance

A practical converter system rarely operates in pure steady-state conditions due to unmeasured and unmodeled disturbances, leading to the modulation index to have small perturbations. In the case that the operating point of a converter system is at a

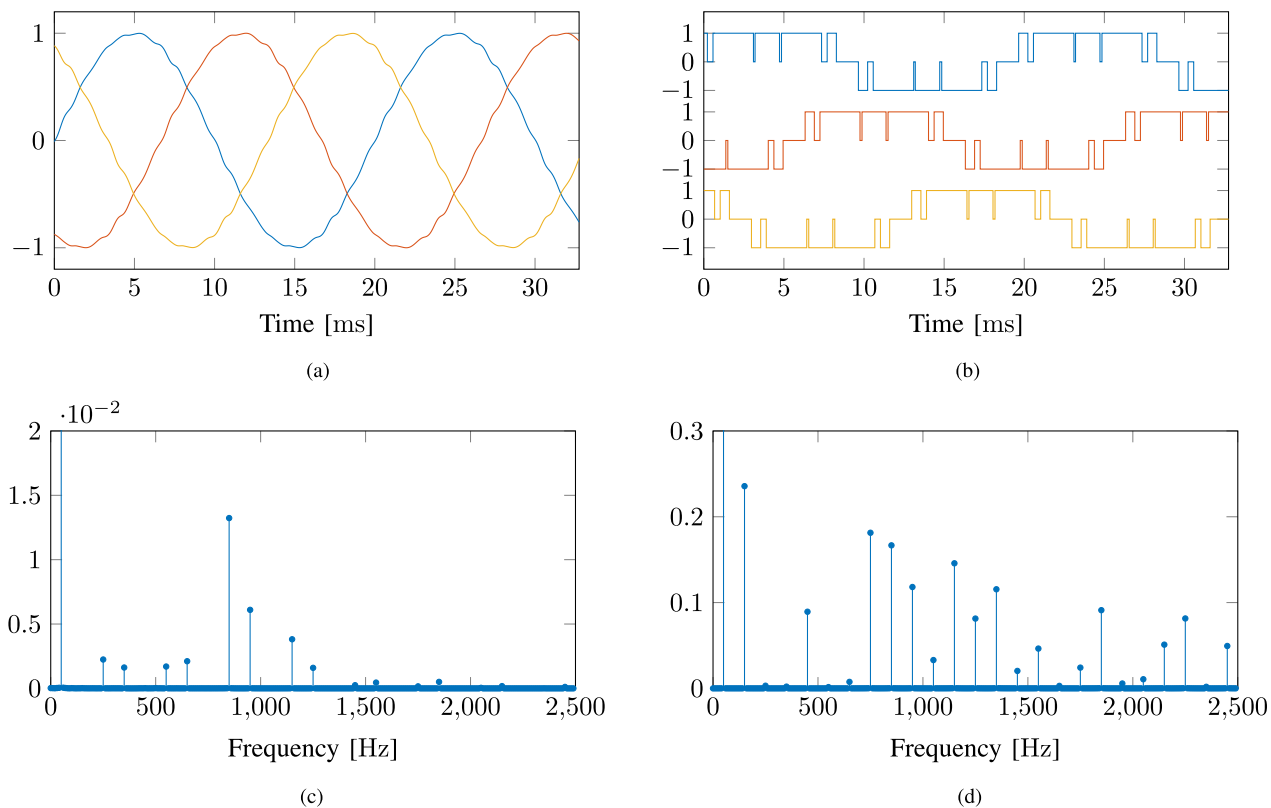


Fig. 7. Steady-state operation. (a) Grid currents. (b) Switch positions. (c) Phase  $a$  grid current spectrum (in pu). (d) Phase  $a$  switching signal spectrum (in pu).

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, which 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 shown 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 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 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<sup>3</sup>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 ms, a symmetrical three-phase-to-ground short-circuit fault occurs and the grid phase voltage drops to 0.05 pu. In order to feed the short circuit, the grid current setpoint is increased to 1.2 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

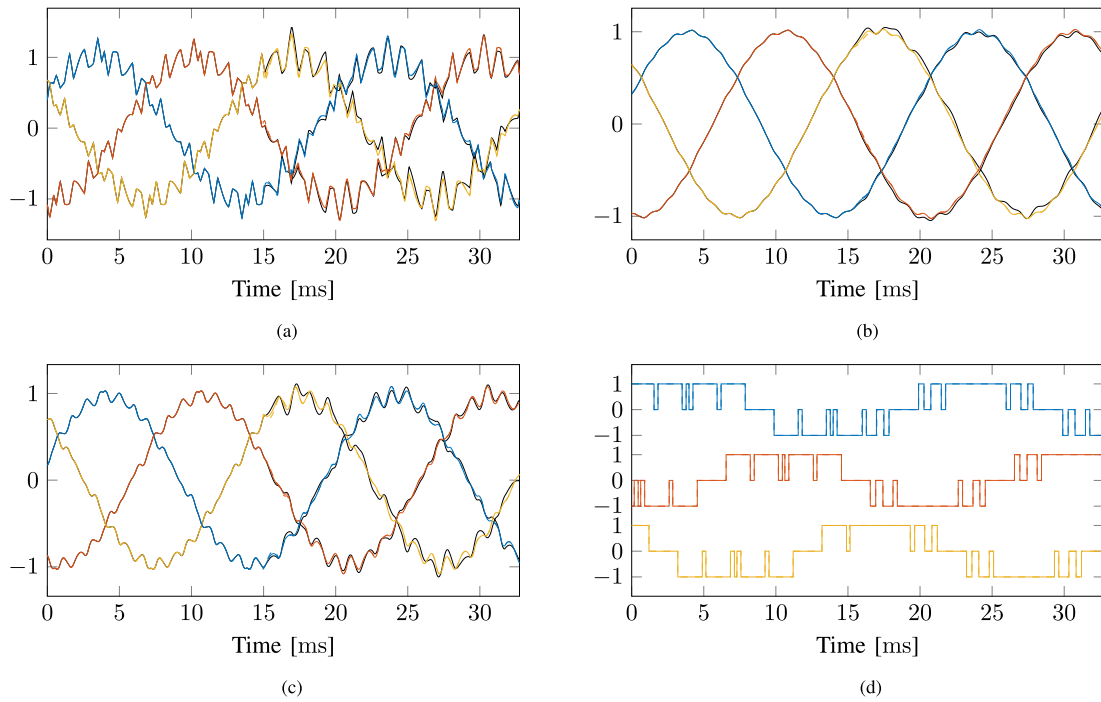


Fig. 8. Quasisteady-state operation. The open-loop responses are shown by the black lines. The steady-state (reference) trajectories are indicated by dashed lines. (a) Converter currents. (b) Grid currents. (c) Capacitor voltages. (d) Switch positions.

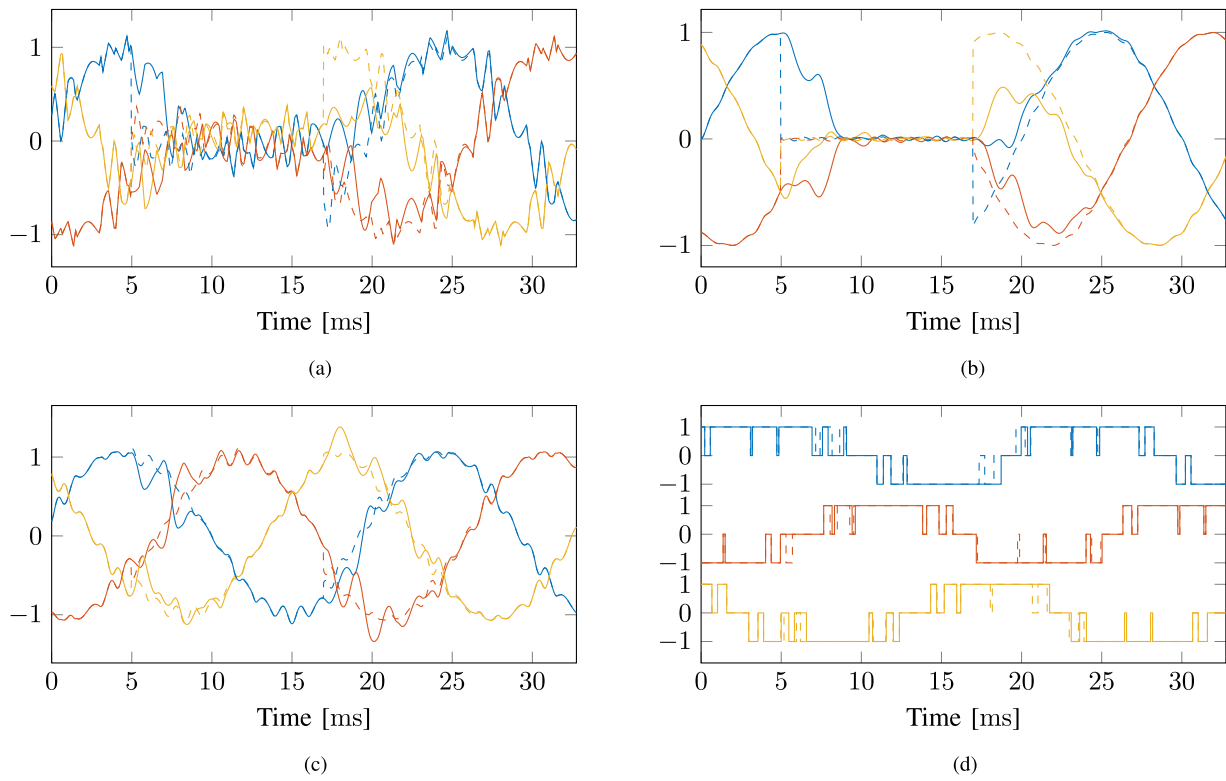


Fig. 9. Reference steps. The steady-state (reference) trajectories are indicated by dashed lines. (a) Converter currents. (b) Grid currents. (c) Capacitor voltages. (d) Switch positions.

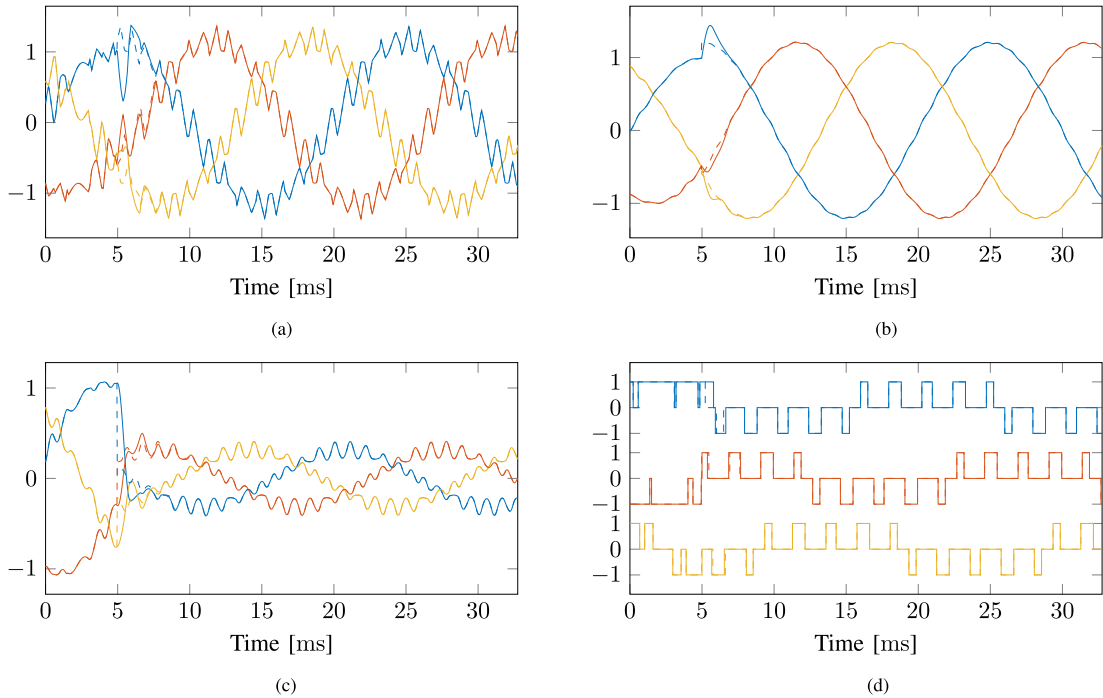


Fig. 10. Grid fault. The steady-state (reference) trajectories are indicated by dashed lines. (a) Converter currents. (b) Grid currents. (c) Capacitor voltages. (d) Switch positions.

loads. By feeding a significant current into the short circuit for up to a few 100 ms, the converter system supports selectivity of the protective devices.

## VI. CONCLUSION

This article focused on the implementation of the QP underlying the proposed controller from [5]. The main challenges were the (relatively) high dimension of 15 of the decision variable, the very short sampling interval of 25  $\mu$ 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 step size of the gradient method can be computed using the Newton's method. Finally, it was explained that 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  $\mu$ 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 the 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  $\mathbf{H}$  and the vector  $\mathbf{c}$  are analyzed and exploited in order to reduce the computational burden with the aim of enabling an efficient implementation on an FPGA.

1) *Exploiting the Problem Structure:* In a first step, to reduce the computational burden, several aspects of the problem can be exploited. First, since  $t_{ij}^* = \max\{t_{p_1,i}^*, t_{p_2,j}^*\}$ , it can be observed that either

$$e^{\mathbf{F}^T(t_{ij}^* - t_{p_1,i}^*)} \quad \text{or} \quad e^{\mathbf{F}(t_{ij}^* - t_{p_2,j}^*)}$$

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

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

where each product may require up to 36 multiplications, is not required to be calculated. For the elements on the diagonal of  $\mathbf{V}$ ,  $i' = j'$  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

$$e^{\mathbf{F}^T(t_{ij}^* - t_{p_1,i}^*)} \quad \text{and} \quad e^{\mathbf{F}(t_{ij}^* - t_{p_2,j}^*)}$$

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

$$\mathbf{V}_{(i',j')} = \mathbf{G}_p^T \Xi (T_p - t_{p,i}^*) \mathbf{G}_p. \quad (22)$$

Finally, note that the term

$$\Xi(T_p - t_{p,i}^*)\mathbf{G}_p$$

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

2) *Exploiting the Load Characteristics*: In the following, characteristics of the load are exploited in order to reduce the computational burden even further. Due to the  $\alpha$  and  $\beta$  components of the system matrix  $\mathbf{F}$  being decoupled, half of the components of the matrix exponential  $e^{\mathbf{F}t}$  are zero; it is of the structure

$$e^{\mathbf{F}t} = \begin{bmatrix} x & 0 & x & 0 & x & 0 \\ 0 & x & 0 & x & 0 & x \\ x & 0 & x & 0 & x & 0 \\ 0 & x & 0 & x & 0 & x \\ x & 0 & x & 0 & x & 0 \\ 0 & x & 0 & x & 0 & x \end{bmatrix}$$

where  $x$  denotes a nonzero element. Thus, the term

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

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  $\mathbf{G}_p$  are nonzero, see (8). This results in matrix–vector multiplications in (6) of the forms

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

only requiring six multiplications (instead of 36).

### B. Matrix Form of Constraints

The matrix  $\mathbf{A}_p$  and the vector  $\mathbf{b}_p$  are defined, respectively, as

$$\mathbf{A}_p =$$

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

and

$$\mathbf{b}_p = \begin{bmatrix} t_{p,1}^* & t_{p,2}^* - t_{p,1}^* & \cdots & T_p - t_{p,n_p}^* \end{bmatrix}^T.$$

### C. Parameters

The parameters of a 9-MVA grid-connected NPC converter with an  $LC$  filter are given in Table 1. 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

TABLE 1  
SYSTEM PARAMETERS OF THE MEDIUM-VOLTAGE CASE STUDY

Parameter	Symbol	Value	PU
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 $\mu$ H	0.0997 pu
Filter capacitance	$C$	420 $\mu$ F	0.1455 pu
Filter inductor ESR	$R$	0.3 m $\Omega$	0.000 27 pu
Filter capacitor ESR	$R_C$	4 m $\Omega$	0.0036 pu
Transformer inductance	$L_t$	526.41 $\mu$ H	0.15 pu
Transformer resistance	$R_t$	16.54 m $\Omega$	0.015 pu
Grid			
Grid-side inductance	$L_g$	349.19 $\mu$ H	0.0995 pu
Grid-side inductor ESR	$R_g$	10.97 m $\Omega$	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

housing. A pu system is established with the following base values:  $V_B = \sqrt{\frac{2}{3}}V_g$ ,  $I_B = \sqrt{2}I_R$ , and  $\omega_B = \omega_1$ .

### ACKNOWLEDGMENT

Tinus Dorfling would like to thank ABB System Drives for their financial support.

### REFERENCES

- [1] G. S. Buja and G. B. Indri, "Optimal pulsewidth modulation for feeding AC motors," *IEEE Trans. Ind. Appl.*, vol. IA-13, no. 1, 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 Trans. Ind. Electron.*, vol. 60, no. 10, 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 Trans. Ind. Informat.*, vol. 9, no. 2, pp. 909–919, May 2013.
- [4] T. Geyer, *Model Predictive Control of High Power Converters and Industrial Drives*. Chichester, U.K.: Wiley, 2016.
- [5] T. Dorfling, H. d. T. Mouton, and T. Geyer, "Generalized model predictive pulse pattern control based on small-signal modelling—Part 1: Algorithm," in *IEEE Trans. Power Electron.*, doi: [10.1109/TPEL.2022.3169113](https://doi.org/10.1109/TPEL.2022.3169113).
- [6] T. Geyer, N. Oikonomou, G. Papafotiou, and F. D. Kieferndorf, "Model predictive pulse pattern control," *IEEE Trans. Ind. Appl.*, vol. 48, no. 2, pp. 663–676, Mar./Apr. 2012.
- [7] J. Nocedal and S. J. Wright, *Numerical Optimization. Springer Series in Operations Research and Financial Engineering*, 2nd ed. New York, NY, USA: Springer, 2006.
- [8] D. P. Bertsekas, *Nonlinear Programming*, 3rd ed. Belmont, MA, USA: Athena Sci., 2016.
- [9] A. Beck, *Introduction to Nonlinear Optimization: Theory, Algorithms, and Applications With MATLAB*. Philadelphia, PA, USA: SIAM, Oct. 2014.
- [10] G. H. Golub and C. F. Van Loan, "Matrix computations," *Johns Hopkins Studies in the Mathematical Sciences*, 4th ed. Baltimore, MD, USA: The Johns Hopkins Univ. Press, p. 780, 2013.
- [11] M. J. Flynn, "On division by functional iteration," *IEEE Trans. Comput.*, vol. C-19, no. 8, pp. 702–706, Aug. 1970.
- [12] R. L. Burden, J. D. Faires, and A. M. Burden, *Numerical Analysis*, 10th ed. Boston, MA, USA: Cengage Learn., 2016.
- [13] S. Richter, T. Geyer, and M. Morari, "Resource-efficient gradient methods for model predictive pulse pattern control on an FPGA," *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 3, pp. 828–841, May 2017.
- [14] A. Németh and S. Németh, "How to project onto an isotone projection cone," *Linear Algebra Appl.*, vol. 433, pp. 41–51, Jul. 2010.

- [15] R. Jasinski, *Effective Coding With VHDL: Principles and Best Practice*. Cambridge, MA, USA: MIT Press, 2016.
- [16] O. Devolder, F. Glineur, and Y. Nesterov, "First-order methods of smooth convex optimization with inexact oracle," *Math. Program.*, vol. 146, pp. 37–75, Aug. 2014.
- [17] N. R. Tavani and V. Dinavahi, "A general framework for FPGA-based real-time emulation of electrical machines for HIL applications," *IEEE Trans. Ind. Electron.*, vol. 62, no. 4, pp. 2041–2053, Apr. 2015.
- [18] T. Geyer and N. Oikonomou, "Model predictive pulse pattern control with very fast transient responses," in *Proc. IEEE Energy Convers. Congr. Expo.*, 2014, pp. 5518–5524.



**Tinus Dorfling** received the B.Eng. degree in electrical and electronic engineering and the M.Eng. and Ph.D. degrees in electrical engineering from the University of Stellenbosch, Stellenbosch, South Africa, in 2015, 2018, and 2021, respectively.

In 2021, he joined ABB Corporate Research Centre, Baden-Dättwil, Switzerland, as a Scientist. His research interest includes modulation and control techniques for medium-voltage converter systems, including optimized pulse patterns, model predictive control, and the implementation of control algorithms on FPGAs.



**Hendrik du Toit Mouton** (Member, IEEE) received the B.Sc., B.Sc. (Hons.), M.Sc., and Ph.D. degrees in mathematics from the University of the Orange Free State, Bloemfontein, South Africa, in 1986, 1987, 1988, and 1991, respectively, and the B.Eng. and Ph.D. degrees in electrical engineering from the University of Stellenbosch, Stellenbosch, South Africa, in 1996 and 2000, respectively.

He is currently a Professor of Electrical Engineering with the University of Stellenbosch and a Leader of Power Electronics Research Group. He

has authored and coauthored more than 130 journal and conference papers in mathematics and power electronics. His research interests include multilevel converters, modeling and control of power electronic converters, and class-D audio amplifiers.



**Tobias Geyer** (Fellow, IEEE) received the Dipl.-Ing. degree in electrical engineering, the Ph.D. degree in control engineering, and the Habilitation degree in power electronics from ETH Zurich, Zurich, Switzerland, in 2000, 2005, and 2017, respectively.

After the Ph.D. degree, he spent three years with GE Global Research, Munich, Germany, three years with the University of Auckland, Auckland, New Zealand, and eight years with ABB's Corporate Research Centre, Baden-Dättwil, Switzerland, where his last position was a Senior Principal Scientist of power conversion control. In 2020, he joined ABB's Medium-Voltage Drive Division, as an R&D Platform Manager of the ACS6080. Working at the intersection of industry and academia, he is currently an extraordinary Professor with Stellenbosch University, Stellenbosch, South Africa. He also teaches a regular course on model predictive control with ETH Zurich. He has authored 35 patent families and the book "Model Predictive Control of High Power Converters and Industrial Drives" (Wiley, 2016). His research interests include medium-voltage and low-voltage drives, utility-scale power converters, optimized pulse patterns, and model predictive control.

Dr. Geyer was the recipient of the Semikron Innovation Award and the Nagamori Award, both in 2021, two Prize Paper Awards of the IEEE transactions, and two Prize Paper Awards at the IEEE conferences. He was an Associate Editor for the *Transactions on Industry Applications* (2011–2014) and the *Transactions on Power Electronics* (2013–2019). He was an International Program Committee Vice Chair of the IFAC conference on Nonlinear Model Predictive Control in Madison, WI, USA, in 2018. He is a Distinguished Lecturer of the Power Electronics Society from 2020 to 2023.