An FPGA Implementation of the Fast Gradient Method for Solving the Model Predictive Pulse Pattern Control Problem

Helfried Peyrl
ABB Corporate Research
5405 Baden-Dättwil, Switzerland
Email: helfried.peyrl@ch.abb.com

Junyi Liu
ABB Corporate Research
5405 Baden-Dättwil, Switzerland
Email: junyi.liu@ch.abb.com

Tobias Geyer
ABB Corporate Research
5405 Baden-Dättwil, Switzerland
Email: tobias.geyer@ch.abb.com

***Abstract*—The computational complexity of Model Predictive Pulse Pattern Control (MP^{3}C) and its real-time requirements have so far limited the deployment of this optimisation-based control method for power electronic systems. In this paper, we present an FPGA implementation, which uses the Fast Gradient Method for solving the quadratic program underlying MP^{3}C. The proposed method can solve the control problem in less than the required 25 µs with sufficient accuracy.**

I. Introduction

During the past decade, model predictive control (MPC) [1], [2] has received significant attention by the power electronics community [3]. Despite the ever increasing computational power available today, MPC is still widely considered to be a control method with too high a computational burden, precluding its applicability to power electronics products. As a consequence, experimental results have been reported so far mostly for simplistic MPC schemes, such as schemes with a prediction horizon of one step [3]. Longer prediction horizons—as shown in simulations—provide significant performance improvements at steady-state operating conditions, but experimental results are scarce [4].

One promising approach to achieve long prediction horizons is to pre-compute offline an optimised pulse pattern (OPP) [5] and to modify it online to achieve fast closed-loop control, adopting the notion of an event-based horizon. We refer to this concept as model predictive pulse pattern control (MP^{3}C) [6]. MP^{3}C has been proposed for controlling induction machines in a variable speed drive setting, by regulating the machine's stator flux vector along an optimal flux trajectory, which is the integral of the OPP. MP^{3}C uses prediction horizons of up to a sixth of the fundamental period and manipulates the time instants of the switching transitions within this horizon, with the objective to eliminate the stator flux error. This formulation results in a mathematical optimisation problem, a so called *quadratic program* (QP), which needs to be solved in real-time. So far, only a simplified version of MP^{3}C based on a deadbeat control approach has been implemented and successfully tested on a 1.1 MVA 5-level medium-voltage inverter drive system [7].

This paper proposes a computational method that solves the QP underlying MP^{3}C in less than 25 µs, thus enabling the use of the quadratic programming version of MP^{3}C for industrial medium-voltage drives. More specifically, we employ the *Fast Gradient Method* (FGM) [8] for solving the QP. The potential of this method for solving input-constrained linear quadratic MPC problems was only recently recognised by Richter et al. in [9]. The main benefits of the FGM are its conceptual simplicity and its low execution times. The recent trend of employing Field Programmable Gate Arrays (FPGAs) for high-speed realisations of MPC [10], [11] inspired the authors of this paper to propose an FPGA implementation of the FGM for solving MPC problems with box-constrained inputs in [12]. In the paper at hand, we present an adaptation of the aforementioned implementation to the specific properties and requirements of MP^{3}C.

II. Problem definition

A. *MP^{3}C problem and principle*

Closed-loop control of an electrical machine based on OPPs can be achieved by controlling the stator flux vector along its reference trajectory [13]. The magnitude of the stator flux trajectory determines the magnetisation current of the machine, while the angle between the stator and the rotor flux linkage vectors determines the electromagnetic torque. At a given time instant, the flux error vector

$$\psi_{s,err} = \psi_s^* - \psi_s$$

is the difference between the reference flux vector ψ_s^* and the actual stator flux vector of the machine ψ_s. The flux vectors and the error vector are visualised in Fig. 1, in which ψ_r denotes the rotor flux vector. The reference of the stator flux trajectory is shown as the dashed (blue) piecewise linear line.

The flux error can be addressed by manipulating the switching instants of the pulse pattern. As an example, consider phase a. With the stator flux being the integral of the voltage applied to the stator windings of the machine, shifting the switching transition by the time Δt_a modifies the stator flux in phase a by

$$\Delta\psi_{sa}(\Delta t_a) = -\frac{V_{\text{dc}}}{2}\Delta u_a \Delta t_a \,,$$

$$V = \frac{V_{\text{dc}}}{6} \begin{bmatrix} 2\Delta u_{a1} & \dots & 2\Delta u_{aN} & -\Delta u_{b1} & \dots & -\Delta u_{bN} & -\Delta u_{c1} & \dots & -\Delta u_{cN} \\ 0 & \dots & 0 & \sqrt{3}\Delta u_{b1} & \dots & \sqrt{3}\Delta u_{bN} & -\sqrt{3}\Delta u_{c1} & \dots & -\sqrt{3}\Delta u_{cN} \end{bmatrix} \quad (3)$$

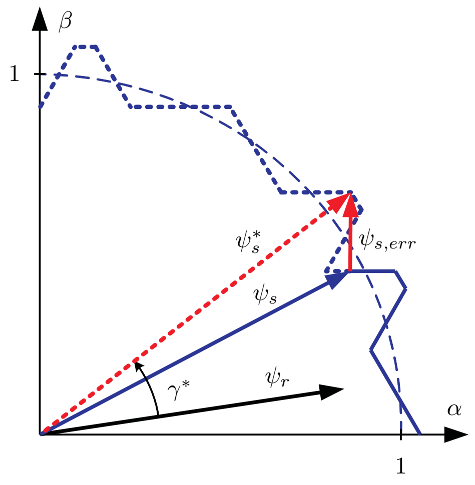

Fig. 1. Stator flux vector ψ_s, reference stator flux vector ψ_s^*, stator flux error vector $\psi_{s,err}$ and rotor flux vector ψ_r in stationary orthogonal coordinates

where V_{dc} refers to the dc-link voltage of the inverter, and $\Delta u_a = u_a(t_a^*) - u_a(t_a^* - \mathrm{d}t)$ denotes the switching transition in phase a, with $\Delta u_a \in \{-1, 1\}$. The nominal switching time is given by t_a^* and $\mathrm{d}t$ is an infinitesimally small time step.

An example is shown in Fig. 2. Delaying the negative switching transition $\Delta u_a = -1$ by Δt_a increases the volt-seconds and thus the stator flux in this phase. Advancing the switching event has the opposite effect, i.e. it decreases the flux amplitude in the direction of phase a. The same holds for phases b and c.

Optimality, i.e. minimal total harmonic distortion (THD) of the stator currents, is achieved when accurately tracking the reference stator flux trajectory. Optimality is thus defined in terms of the reference flux trajectory rather than in terms of the steady-state voltage waveform of the OPP. As a result, the control objective is to regulate the stator flux vector along its reference trajectory, by modifying the switching instants of the OPP within the prediction horizon as little as possible. This results in fast closed-loop control. We refer to this control concept as model predictive pulse pattern control (MP^{3}C).

B. Quadratic programming problem

The MP^{3}C problem is formulated in stationary orthogonal coordinates, with the α and β axes. The stator flux error vector at the discrete time-step k, $\psi_{s,err}(k)$, is the difference between the stator flux reference and the estimated stator flux. In phase a, t_{ai}^*, $i = 1, \dots, N$, denote the nominal switching times of the switching transitions, t_{ai} the corrected times and $\Delta t_{ai} = t_{ai} - t_{ai}^*$ the correction in time. The switching times for phases b and c are defined accordingly. The switching time corrections can be aggregated in the vector $\Delta t = [\Delta t_{a1} \dots \Delta t_{aN} \;\; \Delta t_{b1} \dots \Delta t_{bN} \;\; \Delta t_{c1} \dots \Delta t_{cN}]^T$. The vectors t and t^* are defined in an analogous manner. By manipulating the switching times, a stator flux correction vector $\psi_{s,corr}$ results, which is a linear function of Δt, the dc-link voltage and the switching transitions.

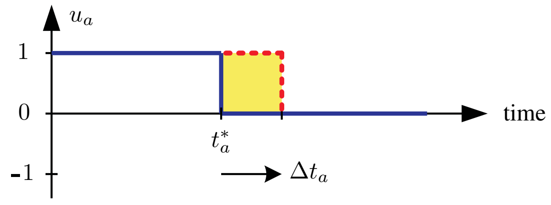

Fig. 2. Delaying the negative switching transition $\Delta u_a = -1$ in phase a by Δt_a, with regard to the nominal switching time t_a^*, increases the stator flux component in this phase by $0.5 V_{\text{dc}} \Delta t_a$

MP^{3}C eliminates the stator flux error by adding an appropriate flux correction vector $\psi_{s,corr}$, while modifying the switching times as little as possible. The latter can be achieved by adding a small penalty Q on Δt. The switching times are linearly constrained by the neighbouring switching times, see Fig. 3.

Using the notation and variables introduced above, the control problem can be recast as the optimisation problem

$$\min_{\Delta t} \; ||\psi_{s,err} - \psi_{s,corr}(\Delta t)||_2^2 + \Delta t^T Q \Delta t \quad (1a)$$
$$\text{s.t. } 0 \le t_{a1} \le t_{a2} \le \dots \le t_{aN} \le t_{a(N+1)}^* \quad (1b)$$
$$0 \le t_{b1} \le t_{b2} \le \dots \le t_{bN} \le t_{b(N+1)}^* \quad (1c)$$
$$0 \le t_{c1} \le t_{c2} \le \dots \le t_{cN} \le t_{c(N+1)}^* \,. \quad (1d)$$

The derivation of (1) is explained in detail in [6].

The stator flux correction in the orthogonal coordinate system can be written as

$$\psi_{s,corr}(\Delta t) = -V \, \Delta t \,, \quad (2)$$

with the voltage matrix defined by (3) at the top of this page.

With (2) the objective function (1a) can be rewritten as

$$J(\Delta t) = (\psi_{s,err} + V \Delta t)^T (\psi_{s,err} + V \Delta t) + \Delta t^T Q \Delta t \,,$$

which can be further simplified, leading to the optimisation problem

$$\min_{\Delta t} \; \Delta t^T (V^T V + Q) \Delta t + 2 \psi_{s,err}^T V \Delta t + \psi_{s,err}^T \psi_{s,err} \quad (4a)$$
$$\text{s.t. (1b)} - \text{(1d)} \,. \quad (4b)$$

After dropping the constant term $\psi_{s,err}^T \psi_{s,err}$, (4) can be posed as a quadratic optimisation problem in the generic form

$$\min_{z} \; f(z) := z^T H z + g^T z \quad (5a)$$
$$\text{s.t. } z + t^* \in \mathcal{Z} \,, \quad (5b)$$

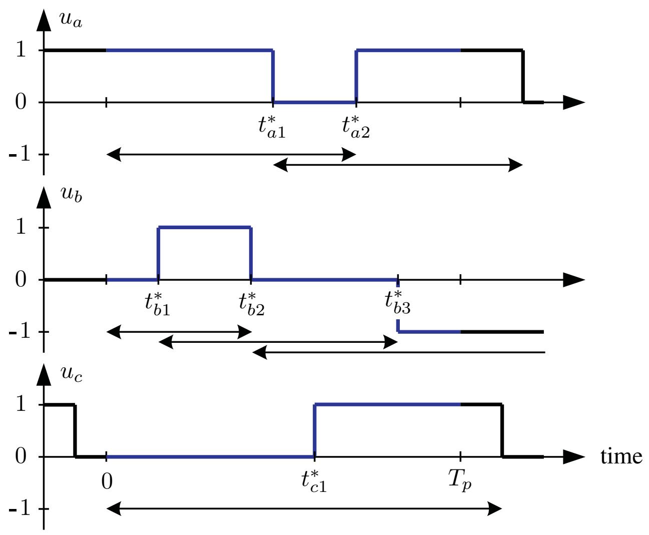

Fig. 3. Model predictive pulse pattern control (MP^{3}C) problem for a three-phase three-level pulse pattern. The lower and upper bounds for the nominal switching instants are depicted by arrows

where $z = \Delta t \in \mathbb{R}^{3N}$, $H = V^T V + Q$ and $g = 2V^T \psi_{s,err}$. The constraint set $\mathcal{Z}$ corresponds to the set described by the inequalities (1b)–(1d). Note that since the matrix V depends on the future switching transitions of the pre-computed pulse pattern, H and g are time-varying and have to be recomputed at every sampling instant.

C. Fast gradient method

The fast gradient method (FGM) [8, Section 2.2.4] is restated here as Algorithm 1. The gradient $\nabla f(z)$ of the

Algorithm 1 Fast gradient method for constrained minimisation

```
Require: Initial point $z^0 \in \mathcal{Z}$, $y^0 = z^0$, number of iterations $i_{max}$, Lipschitz constant $L$, scaling factors $\beta^0, \dots, \beta^{i_{max}-1}$
1: for $i = 0 : i_{max} - 1$ do
2:     $v = y^i - \frac{1}{L}\nabla f(y^i) = y^i - \frac{1}{L}(2Hy^i + g) = My^i + \tilde{g}$
3:     $z^{i+1} = \Pi_{\mathcal{Z}}(v)$
4:     $y^{i+1} = z^{i+1} + \beta^i(z^{i+1} - z^i)$
5: end for
```

quadratic objective function (5a) is given by $2Hz + g$, the operator $\Pi_{\mathcal{Z}}$ in line 3 of Algorithm 1 denotes the orthogonal projection on the constraint set $\mathcal{Z}$. L denotes the Lipschitz constant of the gradient of the objective function (or an upper bound of it). The scaling factors β_i depend on L and a lower bound on the eigenvalues of H, which are computed according to [8]. Since they can be pre-computed, the algorithm is completely division-free (under the assumption that no divisions are required in the projection step) and hence amenable for a fast implementation in hardware.

III. Solving MP^{3}C with the fast gradient method

A key requirement to achieve fast execution times with the FGM is an efficient projection on the constraint set [8]. The feasible set associated with the defining inequalities (1b)–(1d) is usually referred to as a (truncated) monotone order cone. Although there exist efficient algorithms for performing a projection on such a set in higher dimensions (see, e.g. [14]), we choose a different approach: Richter et al. suggested already in their seminal paper [9] to consider *multi-parametric programming* [15] for computing the projection on such a set. The orthogonal projection of a point $\hat{t} \in \mathbb{R}^{3N}$ onto the polyhedral constraint set can be posed as the multi-parametric quadratic program

$$\Pi_{\mathcal{Z}}(\hat{t}) := \arg\min_{t} \; ||\hat{t} - t||_2^2 \qquad \text{s.t. (1b)} - \text{(1d)},$$

with parameters being $\hat{t}$, $t^*_{a(N+1)}$, $t^*_{b(N+1)}$ and $t^*_{c(N+1)}$.

Multi-parametric programming computes the solution of an optimisation problem as a function of its parameters. It was shown in [15] that the optimiser of a multi-parametric quadratic program (MP-QP) is a continuous piecewise-affine function defined over a polyhedral partition of the parameter space. Hence, the projection may be obtained by evaluating

$$t = \begin{cases} E_1 p + e_1 & \text{if } F_1 p \le f_1, \\ \vdots & \\ E_k p + e_k & \text{if } F_k p \le f_k, \end{cases} \tag{6}$$

for some matrices E_i, F_i and vectors e_i and f_i and p denoting the vector of parameters.

The parametric solution (6) can be pre-computed off-line using, for example, the *Multi-Parametric Toolbox (MPT)* for MATLAB [16]. The on-line problem reduces to identifying the polyhedron in which the parameter resides and then to evaluate the associated affine function. One of the most efficient approaches for solving this so-called *point location problem* is based on a pre-computed binary search tree [17]: the method employs the hyperplanes defining the polyhedral regions to build a search tree whose leaves identify univocally the affine map that describes the optimiser. At each parent node of the tree a hyperplane is evaluated and depending on which side the parameter vector resides, the corresponding child node is explored. Finally, each leave node is associated with one of the affine maps $E_i p + e_i$.

The main limitation of multi-parametric programming is the complexity of the solution, which—in the worst case—grows exponentially with the number of parameters [15]. Fortunately, in the case of MP^{3}C, already a small number of transitions (e.g. three) per phase yield an acceptable control performance [6]. Furthermore, the monotonicity constraints on the switching times of different phases are decoupled. Consequently, the multi-parametric solution of the projection can be computed separately for each phase, and it comprises $N + 1$ parameters.

The dimension of the problem can be further reduced by one using a result from Németh et al. who proved in [14] that the orthogonal projection on a truncated monotone order cone can be obtained by first performing a projection on the monotone order cone $\{t \in \mathbb{R}^N \mid t_1 \leq t_2 \leq \cdots \leq t_N\}$ and subsequently setting each coordinate that violates the bounds of the truncated cone to the respective limit.

IV. FPGA implementation

In this section we describe the proposed FPGA implementation of the FGM. It is based on the architecture presented in [12], which we tailored to the particularities of the MP^{3}C problem. The FGM system receives the error in the stator flux vector and the switching times of the pre-computed pulse pattern as inputs, executes the necessary calculations and returns the modified switching times to the modulator.

We employ fixed-point arithmetic because of several reasons: Firstly, if the dynamic range of the numbers is small, processing fixed-point numbers is faster, needs less energy and less area due to the reduced data word size in comparison to floating-point arithmetic [18]. Experimental inspection of the numbers arising during the execution of Algorithm 1 for the MP^{3}C problem confirmed their small dynamic range. Secondly, the projection onto the bounded constraint set ensures that all numbers arising during the execution remain bounded and hence overflows can be easily avoided. Thirdly, fixed-point numbers have the advantage to be processed mostly like integer numbers. Multiplication operations can be mapped efficiently to simple DSP blocks available even on low-cost FPGAs.

A. Description of FPGA units

The block diagram of the proposed architecture is shown in Fig. 4. In the following, we provide a high-level description of the units involved. For more details, we refer the reader to [12] and [19].

1) Finite state machine: The finite state machine generates the necessary sequence of control signals to make all other units work together.

2) Matrix units: In a first step, the Matrix Units (MU) compute the matrices $M = I - \frac{2}{L}(V^T V + Q)$ and $K = -\frac{2}{L}V$. These matrices need to be recomputed at each sampling instant, since V depends on the upcoming switching transitions and is thus time-varying. Since a transition is either positive or negative, the construction of the matrices requires only little arithmetic logic in the FPGA. The matrices are then delivered to the matrix-vector multiplication units (see below).

3) Matrix-vector multiplication unit: The Matrix-Vector Multiplication (MVM) units are responsible for the computation of the vector $\tilde{g} = K\Psi_{s,err}$ and for the gradient descent step $My^i + \tilde{g}$ in line 2 of Algorithm 1. At the beginning of each sampling instant, the MVM units are used to update $\tilde{g}$. Afterwards they compute the gradient steps required in each iteration of the algorithm.

The structure of a single MVM unit is basically an array of $3N$ DSP-like components (Multiply-Accumulate Units (MAC)), one for each row of the matrix. At each cycle a new column of the matrix and the corresponding element of the input vector are fed to the unit, the vector-scalar multiplication is computed and the result accumulated in a register.

Since in MP^{3}C the size of the involved column vectors (and hence the number of required MAC resources) is relatively small compared to the hundreds of hardware DSP blocks available in today's FPGAs, our design allows to use several MVM units in parallel to further accelerate the matrix vector multiplication.

4) Adder tree: The adder tree takes as input the intermediate results of the MVM units and sums them up to obtain the full matrix vector product. Moreover, the adder tree is used to add the nominal switching times t^* to obtain the corrected switching times t needed for the projection unit. With an increasing number of MVM units, pipeline stages must be inserted to avoid the adder tree becoming the critical path, which implies more clock cycles added to the whole computation time. For this reason, there is an architecture trade-off between the speed-up gained via the parallelism in the number of MVM units and the increased delay contributed by the adder tree.

5) Buffer unit: The buffer unit marks the limit between heavy parallelism (MVM) and serial processing pipeline (projection and beta units). Its architecture consists of a shift register that takes the output vector of the adder tree as an input and feeds its elements to the post-processing pipeline.

6) Projection unit: The projection unit executes line 3 of Algorithm 1 and comprises three stages: In the first stage, the decision and leave nodes of the binary search tree are computed in parallel. The second stage is used for multiplexing and scaling the results. In the third stage, upper and lower bounds of the constraints are applied and t^* is subtracted to obtain again the pulse correction times Δt.

7) Beta unit: Finally the beta unit is responsible for performing line 4 of Algorithm 1. The beta-corrected optimisation variables are fed back to the MVM, where the next iteration of the FGM is started.

V. Simulation results

We evaluated the accuracy and computational performance of the FGM and its fixed-point arithmetic implementation in VHDL through simulations. For this, we used a benchmark example, which comprises 2000 instances of the MP^{3}C QP for $N = 3$ switching instants per phase. We employed MATLAB's QP solver `quadprog` to compute reference solutions, which we used to compare our results (obtained from the FGM) against.

Embedded MATLAB was used as hardware programming language because it benefits directly from MATLAB's advanced simulation, testing and debugging capabilities. Each unit of the system is realised as a user-defined function block in Simulink and initiated from MATLAB scripts. Ultimately, we employed MathWorks' HDL Coder for translating the Embedded MATLAB functions into VHDL. The FGM implementation was compiled for the Altera Cyclone V 5CEA7

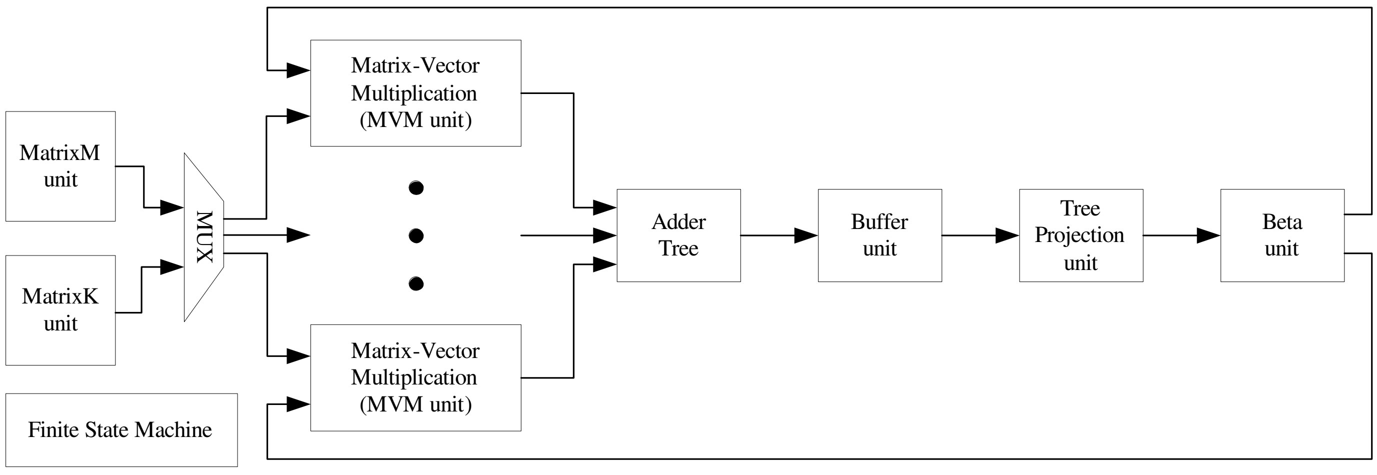

Fig. 4. Block diagram of the proposed FPGA architecture.

using the software tool Quartus II 12.0sp2 Web Edition. The FPGA frequency estimated by the tool was 99 MHz for the design comprising 9 MVM units. In total 2411 cycles are required for 300 FGM iterations which amount to an execution time of 24.4 µs.

In a first experiment, we assessed the effectiveness of the FGM for solving the MP^{3}C problem in comparison to the active-set method of `quadprog`. Fig. 5 compares the absolute error of the objective values of 500 instances for a varying number of FGM iterations using double precision floating point arithmetic. For 300 iterations, the absolute error in the objective values is less then 10^{-8} in all problem instances. The worst-case error in the switching time correction Δt is shown in Fig. 6. With 300 iterations the error in time is less than 10 µs. In the MP^{3}C problem, a high quality of the solution with respect to optimality is needed to ensure a sufficiently small error in the optimiser. This is due to the relatively small weight Q on the time modifications to give high priority to the correction of the stator flux error. Consequently, a suboptimal distribution of the volt-second corrections among the switching times has only a minor impact on the objective value.

Finally, Fig. 7 shows the results obtained with our hardware implementation using fixed-point arithmetic (18 and 27 bits) and compares them with the results obtained using double precision floating point numbers. We used two bits for the integer part which are enough to guarantee that no overflow occurs. The simulation results suggest that for 300 iterations 27 bits are sufficient to obtain solutions of a quality similar to the one obtained with double precision.

VI. Conclusions

Using an event-based horizon, MP^{3}C is a model predictive control method for power electronics, which facilitates long prediction horizons and yields a superior steady-state performance. The mathematical optimisation problem underlying MP^{3}C can be recast as a quadratic program (QP) with input constraints, which needs to be solved in real-time—typically every 25 µs. Due to its conceptual simplicity, fast execution

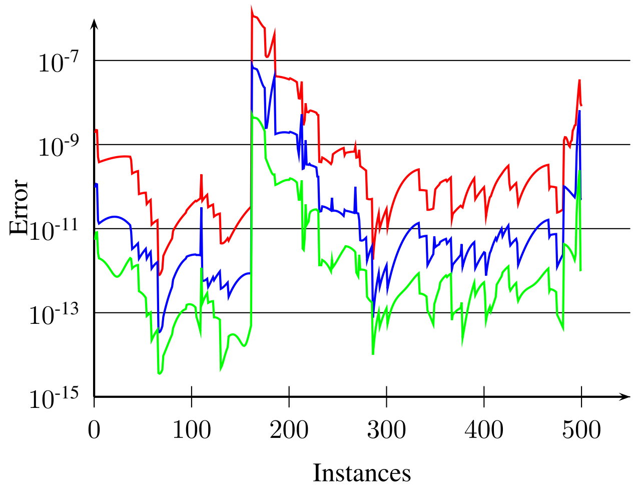

Fig. 5. Absolute error of the objective value for 100 (red), 200 (blue) and 300 (green) FGM iterations using double precision arithmetic.

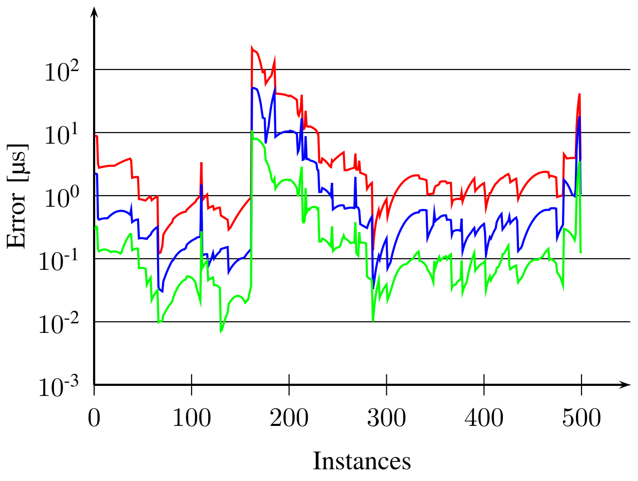

Fig. 6. Maximum errors in the switching times $||\Delta t||_\infty$ for 100 (red), 200 (blue) and 300 (green) FGM iterations using double precision arithmetic.

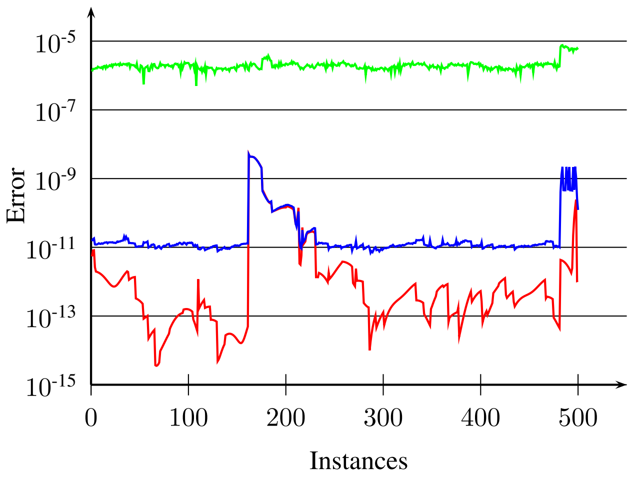

Fig. 7. Absolute error of the objective value for 300 FGM iterations using double precision (red) and 27 bit (blue) and 18 bit (green) fixed-point arithmetic.

times and avoidance of divisions, the fast gradient method (FGM) constitutes a promising approach to solve the QP. As shown in this paper, the FGM can be implemented on an Cyclone V FPGA, solving the QP with a sufficient accuracy in less than the required 25 µs. This enables the use of the full MP^{3}C version, which relies on a QP, rather than the simplified deadbeat version, for high-performance multi-level MV drive systems.

ACKNOWLEDGEMENTS

The authors are grateful to Stefan Richter, Automatic Control Laboratory, ETH Zürich for some very helpful discussions about the fast gradient method and its application to MP^{3}C.

REFERENCES

[1] C. E. Garcia, D. M. Prett, and M. Morari, "Model Predictive Control: Theory and Practice—A Survey," *Automatica*, vol. 25, no. 3, pp. 335–348, Mar. 1989.

[2] J. B. Rawlings and D. Q. Mayne, *Model predictive control: Theory and design*. Madison, WI, USA: Nob Hill Publ., 2009.

[3] P. Cortés, M. P. Kazmierkowski, R. M. Kennel, D. E. Quevedo, and J. Rodríguez, "Predictive control in power electronics and drives," *IEEE Trans. Ind. Electron.*, vol. 55, no. 12, pp. 4312–4324, Dec. 2008.

[4] T. Geyer, "A comparison of control and modulation schemes for medium-voltage drives: emerging predictive control concepts versus PWM-based schemes," *IEEE Trans. Ind. Appl.*, vol. 47, no. 3, pp. 1380–1389, May/Jun. 2011.

[5] G. S. Buja, "Optimum output waveforms in PWM inverters," *IEEE Trans. Ind. Appl.*, vol. 16, no. 6, pp. 830–836, Nov./Dec. 1980.

[6] T. Geyer, N. Oikonomou, G. Papafotiou, and F. Kieferndorf, "Model Predictive Pulse Pattern Control," *IEEE Trans. Ind. Appl.*, vol. 48, no. 2, pp. 663–676, Mar./Apr. 2012.

[7] N. Oikonomou, C. Gutscher, P. Karamanakos, F. Kieferndorf, and T. Geyer, "Model Predictive Pulse Pattern Control for the Five-Level Active Neutral Point Clamped Inverter," in *Proc. IEEE Energy Convers. Congr. Expo.*, Raleigh, NC, USA, Sep. 2012.

[8] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*. Boston: Kluwer Academic Publishers, 2004.

[9] S. Richter, C. N. Jones, and M. Morari, "Real-Time Input-Constrained MPC Using Fast Gradient Methods," in *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, 2009, pp. 7387–7393.

[10] J. L. Jerez, G. A. Constantinides, and E. C. Kerrigan, "FPGA Implementation of an Interior Point Solver for Linear Model Predictive Control," in *Field-Programmable Technology (FPT), 2010 International Conference on*, 2010, pp. 316–319.

[11] A. Wills, A. Mills, and B. Ninness, "FPGA Implementation of an Interior-Point Solution for Linear Model Predictive Control," in *World Congress*, vol. 18, 2011, pp. 14 527–14 532.

[12] M. A. Boéchat, J. Liu, H. Peyrl, A. Zanarini, and T. Besselmann, "An Architecture for Solving Continuous Optimisation Problems on a Field Programmable Gate Array," in *Proc. 21st Mediterranean Conference on Control and Automation*, Crete, 2013.

[13] J. Holtz and N. Oikonomou, "Synchronous Optimal Pulsewidth Modulation and Stator Flux Trajectory Control for Medium-Voltage Drives," *IEEE Trans. Ind. Appl.*, vol. 43, no. 2, pp. 600–608, Mar./Apr. 2007.

[14] A. B. Németh and S. Z. Németh, "How to project onto the monotone nonnegative cone using Pool Adjacent Violators type algorithms," *arXiv:1201.2343*, Jan. 2012. [Online]. Available: http://arxiv.org/abs/1201.2343

[15] B. Bank, J. Guddat, D. Klatte, B. Kummer, and K. Tammer, *Non-Linear Parametric Optimization*. Berlin: Akademie-Verlag, 1982.

[16] M. Kvasnica, P. Grieder, M. Baotić, and M. Morari, "Multi Parametric Toolbox (MPT)," in *HSCC (Hybrid Systems: Computation and Control)*, ser. Lecture Notes in Computer Science, R. Alur and G. Pappas, Eds. Springer Verlag, 2004, vol. 2993, pp. 448–462.

[17] P. Tøndel, T. A. Johansen, and A. Bemporad, "Evaluation of piecewise affine control via binary search tree," *Automatica*, vol. 39, no. 5, pp. 945–950, 2003.

[18] W. T. Padgett, D. V. Anderson, and J. Moura, *Fixed-Point Signal Processing*. Morgan and Claypool Publishers, Sept. 2009.

[19] M. A. Boéchat, "High-Speed Model Predictive Control Using the Fast Gradient Method on FPGA," Master thesis, EPFL, ABB Corporate Research, 2012.