Resource-Efficient Gradient Methods for Model Predictive Pulse Pattern Control on an FPGA

Stefan Richter, Tobias Geyer, *Senior Member, IEEE,* and Manfred Morari, *Fellow, IEEE*

***Abstract*—We demonstrate that an optimization-based Model Predictive Pulse Pattern Controller (MP^{3}C) can be designed with a complexity in terms of hardware resource usage on a field programmable gate array (FPGA) that is comparable to that of a conventional controller. Key to the superior performance and resource usage over existing solution methods for MP^{3}C are an appropriate problem reformulation and a newly derived result for the projection on the truncated monotone cone that composes the feasible set in this application. Using a coldstarted classic gradient method in fixed-point arithmetic, a numerically stable implementation is shown to require less than 300 clock cycles to meet the stringent accuracy specification for problems with at most three switching transitions per phase. For the case of four (five) transitions, only about 550 (690) cycles are required. At the same time, merely two DSP-type multipliers on an FPGA are used for all problem sizes. These results indicate a speed improvement of ten times and a resource reduction by 17 times in terms of DSP-type multipliers for the case of three transitions per phase when compared to an existing solution method. When there are four or five transitions per phase, the resource reductions are even more impressive.**

***Index Terms*—Embedded optimization, model predictive control, optimized pulse patterns, gradient methods, FPGA.**

I. Introduction

THE trade-off between harmonic current distortions and the switching frequency at which the power semiconductors are operated is fundamental to the field of power electronics. Optimized pulse patterns (OPPs) provide the optimal trade-off by minimizing the harmonic current distortions for a given switching frequency [1]. They are particularly attractive when operating the converter at low ratios between switching and fundamental frequency [2]. Examples include traction and medium-voltage converters used for variable speed drives [3].

Under idealized conditions, OPPs provide the optimal steady-state performance for power electronics converters. In reality, however, fluctuations in the DC-link voltage, converter nonlinearities and the need to switch between different OPPs severely impact their performance. To mitigate these issues and to achieve dynamic control of the load currents, it is mandatory to augment OPPs by a fast closed-loop controller.

In power electronics, pulse width modulation schemes are traditionally used that feature fixed-length modulation intervals. Regularly-spaced sampling instants exist, at which the current ripple is zero. This allows one to neglect the switching ripple and to adopt the concept of averaging, based on which linear current control loops can be designed. As a result, PI current controllers are almost exclusively used in industrial power electronics converters [4].

To minimize the current distortions in OPPs, the concept of fixed-length modulation intervals is relinquished, making it intrinsically hard to achieve fast closed-loop control based on OPPs using traditional PI-based control concepts [5]. When considering a three-phase induction machine fed by a voltage source inverter, fast current control can be achieved by regulating the machine's stator flux linkage. The stator flux is the time integral of the applied stator voltage, which is, under nominal conditions, equal to the OPP switching waveform multiplied with the DC-link voltage. This leads to an optimal stator flux trajectory of the OPP. Fast machine control based on OPPs can then be achieved by regulating the stator flux along this optimal trajectory by manipulating the time instants of the switching transitions [6]. To this end, a deadbeat flux controller can be designed [7].

The notion of deadbeat trajectory tracking was generalized in [8]. The flux error correction and the manipulation of the switching instants is performed within a prediction horizon of a fixed length. The impact of the switching time modifications is predicted using a model, albeit a very simple one, and the receding horizon policy is adopted. This leads to an OPP-based model predictive controller (MPC) [9], [10], to which we refer as Model Predictive Pulse Pattern Control (MP^{3}C) [8]. The optimization problem underlying Model Predictive Pulse Pattern Control (MP^{3}C) is a quadratic program (QP) with a decision vector that is of a time-varying dimension.

MP^{3}C is an attractive control method for high-power converters operating at low switching frequencies. MP^{3}C combines the dynamic performance of a high-bandwidth controller with the superb harmonic performance of OPPs during steady-state operation. Solving the QP in real time, however, has proven to be a highly challenging task, given the short sampling intervals of 25 µs typically used and the prevalence of low-cost low-performance field programmable gate arrays (FPGAs) which serve as computational platforms.

A. *MP^{3}C Problem Setup*

In the following, we give a formal description of the MP^{3}C problem as a QP. In the stationary and orthogonal α-β coordinate system the problem can be written as

$$\begin{aligned}\min_{\Delta t}\ & \frac{1}{2}\|\psi_{\text{s,err}} - \psi_{\text{s,corr}}(\Delta t)\|^2 + \frac{q}{2}\|\Delta t\|^2 \\ \text{s.t. }\ & \Delta t \in \mathbb{X} - \bar{t}\,,\end{aligned} \tag{1}$$

S. Richter is with Richter Optimization GmbH, 8046 Zürich, Switzerland (e-mail: sr@richteroptimization.com).

T. Geyer is with ABB Corporate Research, Baden-Dättwil, Switzerland.

M. Morari is with the Automatic Control Laboratory, ETH Zürich, 8092 Zürich, Switzerland.

where $\psi_{\text{s,err}}$ is the stator flux error and $\psi_{\text{s,corr}}(\Delta t)$ is the stator flux correction. The decision vector Δt is composed of all modifications from the OPP's nominal switching times $\bar{t}$ in phases 'a', 'b' and 'c', i.e. $\Delta t = t - \bar{t}$ where $\Delta t = (\Delta t_{\text{a}}, \Delta t_{\text{b}}, \Delta t_{\text{c}})$, $t = (t_{\text{a}}, t_{\text{b}}, t_{\text{c}})$ and $\bar{t} = (\bar{t}_{\text{a}}, \bar{t}_{\text{b}}, \bar{t}_{\text{c}})$[1]. Despite the fixed-length prediction horizon, the number of switching transitions per phase and hence the problem size of the QP in (1) is varying. However, there is at least a single transition and a maximum of n transitions per phase. Therefore, the length of the decision vector can take values from $\{3, 4, \ldots, 3n\}$. In practice, longer prediction horizons and thus higher values of n render the MP^{3}C controller more robust against measurement noise [8]. Note that typical values for n are from $\{3, 4, 5\}$. From here on we let $n_{\text{a}}, n_{\text{b}}$ and n_{c} ($\leq n$) denote the *actual* number of switching transitions in the corresponding phase within the prediction horizon.

By modifying the switching times of the OPP, the stator flux error $\psi_{\text{s,err}} \in \mathbb{R}^2$, which is the difference between the reference and the estimated stator flux, can be corrected over the prediction horizon. Indeed, the flux correction $\psi_{\text{s,corr}}$ is linear in Δt and is given by

$$\psi_{\text{s,corr}}(\Delta t) = -\frac{V_{\text{dc}}}{6} \cdot \begin{bmatrix} 2 & -1 & -1 \\ 0 & \sqrt{3} & -\sqrt{3} \end{bmatrix} \cdot N \cdot \Delta t\,, \qquad (2)$$

with the matrix

$$N = \begin{bmatrix} \Delta u_{\text{a}}^T & 0^T & 0^T \\ 0^T & \Delta u_{\text{b}}^T & 0^T \\ 0^T & 0^T & \Delta u_{\text{c}}^T \end{bmatrix}, \qquad (3)$$

whose zero row vectors are of appropriate dimensions. The switch position changes associated with the switching transitions for phase 'a' are denoted by the vector Δu_{a}. Since there are only positive or negative switching transitions, we have $\Delta u_{\text{a}} \in \{-1, 1\}^{n_{\text{a}}}$. The vectors Δu_{b} and Δu_{c} are defined accordingly. Note that all of these vectors are determined by the OPP and so are additional parameters of the MP^{3}C problem. The DC-link voltage V_{dc} also varies in practice but will be assumed constant in this paper.

Solving the MP^{3}C problem in (1) provides a tradeoff between flux error correction and modification from the nominal OPP. The tradeoff can be adjusted by the scalar weight $q > 0$. The set constraint $\Delta t \in \mathbb{X} - \bar{t}$ ensures feasibility of the optimal switching times $t^* = \Delta t^* + \bar{t}$. In fact, the set $\mathbb{X}$ is defined as $\mathbb{X} = \mathbb{X}_{\text{a}} \times \mathbb{X}_{\text{b}} \times \mathbb{X}_{\text{c}}$, where, e.g. for phase 'a'

$$\mathbb{X}_{\text{a}} = \left\{ t_{\text{a}} \in \mathbb{R}^{n_{\text{a}}} \,\middle|\, 0 \leq t_{\text{a}_1} \leq t_{\text{a}_2} \leq \ldots \leq t_{\text{a}_{n_{\text{a}}}} \leq \bar{t}_{\text{a}_{n_{\text{a}}+1}} \right\}. \quad (4)$$

This definition ensures that the optimal switching times happen in the future in ascending order but prior to the $(n_{\text{a}} + 1)$th transition with the nominal transition time $\bar{t}_{\text{a}_{n_{\text{a}}+1}}$.

In summary, the MP^{3}C problem is a small-scale QP of varying size with the parameters $\psi_{\text{s,err}}, \bar{t}, \bar{t}_{\text{a}_{n_{\text{a}}+1}}, \bar{t}_{\text{b}_{n_{\text{b}}+1}}$ and $\bar{t}_{\text{c}_{n_{\text{c}}+1}}$ along with $\Delta u_{\text{a}}, \Delta u_{\text{b}}$ and Δu_{c}.

B. Requirements on the Solution Method

We briefly summarize the specifications any solution method must meet. Note that we consider an FPGA as the computational platform. This is the most prevalent platform on medium-voltage converters due to the short, deterministic execution times and the ability to parallelize computations. Although the content of this section also applies to platforms such as (serial) digital signal processors (DSPs), the requirements on hardware resources below mainly concern FPGAs.

a) Solution Time: The sampling interval is 25 µs. Since MP^{3}C represents only a part of the overall controller structure (cf. Fig. 7 in [8]) and other blocks need to be evaluated too, only about 20 µs are available to derive a solution. On an FPGA with 40 MHz clock rate, for example, this amounts to 800 cycles.

b) Accuracy: Simulations indicate that closed-loop performance using a feasible, suboptimal solution Δt satisfying $\|\Delta t - \Delta t^*\|_\infty \leq 10$ µs is sufficiently close to the performance obtained with the optimal solution Δt^*. In view of horizon lengths of up to 5 ms, this means a tight accuracy requirement.

c) Hardware Resource Usage: The solution method needs to be implemented in fixed-point arithmetic since it is significantly more efficient in terms of required programmable logic blocks when compared to floating-point arithmetic. 'Cheap' and fast operations in fixed-point arithmetic with respect to resource usage and latency are additions, subtractions, negations, comparisons and multiplications/divisions by powers of two (the latter amount to left and right shifts, respectively). On most platforms, these operations have a latency of one clock cycle. All other multiplications are considered fast only if they are mapped on DSP blocks, in which case the product can be calculated in a single cycle too (plus two cycles to initialize the input and output registers). Divisions, square roots and more sophisticated functions are computed iteratively or are derived from lookup tables and are thus considered 'expensive' and slow. In general, multiple clock cycles are required for these operations.

The number of available DSP blocks on small, low-cost FPGAs is typically in the order of tens. Furthermore, such FPGAs operate on 18 bit words only, i.e. if numbers are represented with more than 18 bits, multiple DSP blocks need to be reserved for the evaluation of a single product. As we want to explore the possibility of running MP^{3}C on a low-cost FPGA, we can only afford using a few DSP-type multipliers. In contrast, high-performance FPGAs feature hundreds of DSP blocks and 27 bit words.

d) Avoidance of Overflow: Fixed-point numbers are uniquely defined by their integer and fractional part (and the sign, if applicable). For the solution method to be safely implementable, an upper bound on the largest absolute number in the solution method must be known *a priori* so that the number of bits for the integer part can be determined and overflows avoided. Of course, this upper bound can be deduced from extensive numerical simulations. Yet, an analytic bound that covers all possible scenarios is preferred.

The above list of requirements will guide the design of the solution method. It will be shown that all specifications are met by our proposed method. Taking into account resource constraints at the design stage is an example of *co-designing* a solution method along with its implementation. For further aspects on co-design we refer the interested reader to [11].

[1]We use the notation (x, y) to abbreviate $[x^T, y^T]^T$ in this paper.

C. Review of Existing Solution Methods for MP^{3}C

So far, existing solution methods only fulfill a subset of the preceding stringent requirements. The original paper on MP^{3}C [8] suggests two *approximate* solution methods, both meeting the requirements on resource usage and overflow but sacrificing accuracy. The first approach is a primal active set method that repeatedly solves an unconstrained subproblem followed by a clipping step that ensures feasibility. This approach can handle changing problem sizes, since the subproblem has constant size independent of the number of switching transitions. It was found from extensive simulations that only very few, i.e. 2 to 3, iterations are necessary to compute the optimal solution. However, there is no proof yet that rigorously guarantees this for all scenarios and all problem sizes.

In the same paper, a computationally even cheaper approach based on deadbeat control is introduced. An actual implementation of this approach on a medium-voltage drive is presented in [12]. The trade-in for being able to meet the resource requirements is the modification of the QP in (1). Specifically, switching time modifications are not penalized ($q = 0$) and the prediction horizon is determined online as the minimal horizon that spans switching transitions in two phases. Consequently, the flux error correction is limited to modifying the switching times in two phases only, which makes the control susceptible to measurement noise.

The first *exact* solution method for the MP^{3}C problem is reported in [13]. In order to slightly simplify the solution method, the original QP in (1) is altered so that at every sampling instant there are exactly n switching transitions per phase, thus circumventing the problem of changing problem sizes. The solution method builds upon Nesterov's fast gradient method [14] and is guaranteed to converge to an optimal solution in infinite precision (hence, we call it an *exact* solution method). In finite precision, particularly in fixed-point arithmetic, [13] proves that the error still remains bounded and that for the maximum number of switching transitions per phase of $n = 3$ the accuracy requirement is met after about 2400 clock cycles for word lengths of 27 bits. In order to attain this speed, more than 90 DSP-type multipliers are used to fully parallelize the fast gradient method. As follows from these numbers, an implementation on a low-cost FPGA is not achievable[2].

In short, no exact solution method for the original MP^{3}C problem in (1) meeting all the requirements of Section I-B has been proposed so far. To show that this is indeed possible is the purpose of this paper.

Before outlining the contents of this paper, we want to point out two more aspects regarding the fast and resource-efficient solution of MPC-related optimization problems such as MP^{3}C. An alternative and popular approach for the implementation of optimization-based controllers is the explicit approach based on multiparametric programming [15] (see, e.g., [16] for an FPGA implementation in a DC-DC power converter). Since the Hessian in MP^{3}C depends on the switching transitions Δu_a, Δu_b and Δu_c and multiparametric programming is tractable only for nonparametric Hessians, an explicit solution needs to be computed for every possible Hessian. For the case of $n = 3$, for example, there are 864 different Hessians and each of them has an associated explicit solution with about 1500 to 2500 polyhedral partitions of the 14-dimensional parameter space as computations with the MPT3 toolbox [17] have shown. It follows that for MP^{3}C an approach using the explicit solution is intractable even on state-of-the-art FPGAs.

Finally, we want to point out that research on efficient optimization methods and implementations is very active at the time being, e.g. [18], [19]. A very recent overview of other relevant work in this field can be found in [20].

D. Contribution and Outline of the Paper

In Section II we introduce a constant-size reformulation of the QP in (1). This allows us to operate on constant memory and resources, independent from the actual number of transitions per phase. This is a prerequisite for an efficient implementation. We will then reformulate the primal problem as a dual problem by first introducing new variables and constraints and then dualizing them. The resulting dual problem exhibits much better convergence behavior than the primal problem when solved with gradient methods.

Section III gathers all pieces required to derive an algorithmic solution to the dual MP^{3}C problem. First, we summarize the basics of the gradient methods considered in this paper, in particular, the classic gradient method and Nesterov's fast gradient method (Section III-A). After that, we map these methods to the dual of the MP^{3}C problem (Section III-B) and provide an in-depth treatment of the key issues for resource-efficient and fast implementations. These issues include the projection on the feasible set (Section III-B1), preconditioning (Section III-B2) and the optimal choice of the step size enlargement factor for the classic gradient method (Section III-B3). The feasible set in MP^{3}C is composed of (shifted) *truncated monotone cones* (cf. set $\mathbb{X}_\text{a}$ in (4)). We present a new result for the projection on this set that will enable us to arrive at implementations with a very small resource footprint. In fact, we will show that for any dimension of this cone, the projection can be computed without using any DSP-type multiplier. With respect to preconditioning, we prove that there does not exist a static preconditioner in the dual domain that can decrease the worst case condition number over all possible combinations of switching transitions.

Further fixed-point aspects come into play in Section IV. The most important aspect is to avoid overflow errors. Section IV-A presents an analysis that derives an upper bound on the largest absolute number that can occur inside the gradient methods, and which is valid for all possible scenarios. The aforementioned analysis will lead to guidelines on how to scale the dual iterates so that a high accuracy with short bit lengths can be accomplished (Section IV-B). If done carefully, the appropriate scaling of iterates can also facilitate the reduction

[2]In fact, trading in more solution time for less resources is doable in the approach reported in [13]. Specifically, an implementation with 34 DSP-type multipliers taking about 3000 clock cycles can be realized. Similar implementations for problem instances with $n \in \{4, 5\}$ would require 43 and 52 DSP-type multipliers, respectively (from personal communication with the main author of [13]). These numbers serve as a basis for the comparison with our implementation.

of the required DSP-type multipliers. Section IV-C concludes with an estimate of the number of clock cycles for a single iteration of the used gradient methods.

Section V presents fixed-point simulation results proving that the classic gradient method meets all of the requirements in Section I-B for problem sizes up to $n = 5$ transitions per phase. Finally, Section VI gives an outlook for potential future work.

II. Dual Constant-Size Reformulation

For a resource-efficient implementation it is of first and foremost importance that the problem size remains constant. In order to achieve this, we embed all size-varying instances of the MP^{3}C problem in (1) into the equivalent reformulation

$$\begin{aligned}\min_{\Delta t_\mathrm{r}}\ & \frac{1}{2}\|\psi_\text{s,err} - \tilde{\psi}_\text{s,corr}(\Delta t_\mathrm{r})\|^2 + \frac{q}{2}\|\Delta t_\mathrm{r}\|^2 \\ \text{s.t. }\ & \Delta t_\mathrm{r} \in \mathbb{X}_\mathrm{r} - \bar{t}_\mathrm{r}\end{aligned} \tag{5}$$

with the decision vector Δt_r that is of the constant length $3n$.

The flux error correction $\tilde{\psi}_\text{s,corr}$ is defined according to (2), but the size-varying matrix N is now replaced by the constant-size matrix N_r given as

$$N_\mathrm{r} = \begin{bmatrix} (\Delta u_\mathrm{a}, 0_{n-n_\mathrm{a}})^T & 0^T & 0^T \\ 0^T & (\Delta u_\mathrm{b}, 0_{n-n_\mathrm{b}})^T & 0^T \\ 0^T & 0^T & (\Delta u_\mathrm{c}, 0_{n-n_\mathrm{c}})^T \end{bmatrix},$$

where the zero vector of length m is denoted as 0_m. In the constant-size reformulation, the feasible set is determined by $\mathbb{X}_\mathrm{r} = \mathbb{X}_\text{r,a} \times \mathbb{X}_\text{r,b} \times \mathbb{X}_\text{r,c}$, and $\bar{t}_\mathrm{r} = (\bar{t}_\text{r,a}, \bar{t}_\text{r,b}, \bar{t}_\text{r,c})$ is defined as

$$\begin{aligned}\mathbb{X}_\text{r,a} &= \left\{t_\text{r,a} \in \mathbb{R}^n \mid 0 \le t_{\text{r,a}_1} \le \ldots \le t_{\text{r,a}_n} \le \bar{t}_{\text{a}_{n_\text{a}+1}}\right\}, \\ \bar{t}_\text{r,a} &= \left(\bar{t}_\text{a},\, \bar{t}_{\text{a}_{n_\text{a}+1}} \cdot 1_{n-n_\text{a}}\right)\end{aligned} \tag{6}$$

for phase ‘a’ and analogously for phases ‘b’ and ‘c’.

In order to see that every solution to the original MP^{3}C problem in (1) can be recovered from the solution to the reformulation in (5), we let

$$\Delta t_\mathrm{r} = \left((\Delta t_\mathrm{a}, \overline{\Delta t_\mathrm{a}}), (\Delta t_\mathrm{b}, \overline{\Delta t_\mathrm{b}}), (\Delta t_\mathrm{c}, \overline{\Delta t_\mathrm{c}})\right)$$

with, e.g., $\overline{\Delta t_\mathrm{a}}$ being an auxiliary vector of length $n - n_\mathrm{a}$, so that all auxiliary vectors together with the original modification vectors ensure a constant length of the vector Δt_r. By means of partial minimization, the reformulation can be rewritten as

$$\begin{aligned}\min_{\Delta t = (\Delta t_\mathrm{a}, \Delta t_\mathrm{b}, \Delta t_\mathrm{c})}\ & \frac{1}{2}\|\psi_\text{s,err} - \psi_\text{s,corr}(\Delta t)\|^2 + \frac{q}{2}\|\Delta t\|^2 + \\ & \min_{\overline{\Delta t_\mathrm{a}}, \overline{\Delta t_\mathrm{b}}, \overline{\Delta t_\mathrm{c}}} \frac{q}{2}\left(\|\overline{\Delta t_\mathrm{a}}\|^2 + \|\overline{\Delta t_\mathrm{b}}\|^2 + \|\overline{\Delta t_\mathrm{c}}\|^2\right) \\ & \quad \text{s.t. } (\Delta t_\mathrm{a}, \overline{\Delta t_\mathrm{a}}) \in \mathbb{X}_\text{r,a} - \bar{t}_\text{r,a} \\ & \qquad\ \ (\Delta t_\mathrm{b}, \overline{\Delta t_\mathrm{b}}) \in \mathbb{X}_\text{r,b} - \bar{t}_\text{r,b} \\ & \qquad\ \ (\Delta t_\mathrm{c}, \overline{\Delta t_\mathrm{c}}) \in \mathbb{X}_\text{r,c} - \bar{t}_\text{r,c}\,.\end{aligned}$$

For every feasible modification in the original problem setup, $\Delta t \in \mathbb{X} - \bar{t}$, the inner minimization problem has an optimal value of zero, whereas for every infeasible modification, $\Delta t \notin \mathbb{X} - \bar{t}$, the inner minimization problem is infeasible. Hence, the optimal solution of the reformulation is

$$\Delta t_\mathrm{r}^* = \left((\Delta t_\mathrm{a}^*, 0_{n-n_\mathrm{a}}), (\Delta t_\mathrm{b}^*, 0_{n-n_\mathrm{b}}), (\Delta t_\mathrm{c}^*, 0_{n-n_\mathrm{c}})\right),$$

where $\Delta t^* = (\Delta t_\mathrm{a}^*, \Delta t_\mathrm{b}^*, \Delta t_\mathrm{c}^*)$ is the optimal solution to the size-varying original problem. As a result, we can recover the optimal size-varying modifications Δt^* to the original problem from the optimal solution to the constant-size reformulation. Note that the reformulation is easily implementable on an FPGA since only cheap operations are required for its setup.

Empirical evidence shows that gradient methods applied in the primal domain, i.e. the domain in which the MP^{3}C problem is formulated, have a slow convergence. This is due to tight accuracy requirements (cf. Section I-B) and the bad conditioning as a consequence of a small weight q[3]. The authors of [13] report that for the case of $n = 3$ there are 300 iterations necessary for the fast gradient method to achieve the required accuracy level on a slightly different problem setup than used in this paper. Preliminary work has shown similar iteration counts for our constant-size reformulation and also that optimal preconditioning by means of diagonal preconditioner matrices only marginally improves the conditioning (apart from unacceptable memory requirements for storing these matrices for *every* possible Hessian matrix).

Interestingly, convergence is much faster if the problem is solved in an appropriate dual domain, although the condition number turns out to be of the same order as in the primal domain. For this, we first introduce a new primal variable $\tilde{\psi}_\text{s,corr}$ for the flux error correction (by a slight abuse of notation) so that problem (5) becomes

$$\begin{aligned}\min_{\Delta t_\mathrm{r}, \tilde{\psi}_\text{s,corr}}\ & \frac{1}{2}\|\psi_\text{s,err} + \tilde{\psi}_\text{s,corr}\|^2 + \frac{q}{2}\|\Delta t_\mathrm{r}\|^2 \\ \text{s.t. }\ & \tilde{\psi}_\text{s,corr} = V_\mathrm{r}\Delta t_\mathrm{r} \\ & \Delta t_\mathrm{r} \in \mathbb{X}_\mathrm{r} - \bar{t}_\mathrm{r}\,,\end{aligned}$$

with the matrix

$$V_\mathrm{r} = \frac{V_\text{dc}}{6} \cdot \begin{bmatrix} 2 & -1 & -1 \\ 0 & \sqrt{3} & -\sqrt{3} \end{bmatrix} \cdot N_\mathrm{r}\,. \tag{7}$$

If we dualize the new equality constraint by means of the dual multiplier $\lambda \in \mathbb{R}^2$, i.e.

$$\tilde{d}(\lambda) = \min_{\Delta t_\mathrm{r} \in \mathbb{X}_\mathrm{r} - \bar{t}_\mathrm{r}} \tfrac{1}{2}\|\psi_\text{s,err} + \tilde{\psi}_\text{s,corr}\|^2 + \tfrac{q}{2}\|\Delta t_\mathrm{r}\|^2 + \lambda^T(\tilde{\psi}_\text{s,corr} - V_\mathrm{r}\Delta t_\mathrm{r}),$$

we obtain the unconstrained dual problem in minimization form

$$\min_\lambda d(\lambda) \tag{8}$$

with the convex dual function $d(\lambda) \triangleq -\tilde{d}(\lambda)$ given as

$$d(\lambda) = \frac{1}{2}\|\lambda\|^2 + \psi_\text{s,err}^T\lambda + \max_{\Delta t_\mathrm{r} \in \mathbb{X}_\mathrm{r} - \bar{t}_\mathrm{r}} -\frac{q}{2}\|\Delta t_\mathrm{r}\|^2 + \lambda^T V_\mathrm{r}\Delta t_\mathrm{r}\,. \tag{9}$$

It can be shown that the primal solution Δt_r^* can be recovered from the inner maximization problem given the optimal dual solution or so-called Lagrange multiplier λ^*, i.e.

$$\Delta t_\mathrm{r}^* = \arg\max_{\Delta t_\mathrm{r} \in \mathbb{X}_\mathrm{r} - \bar{t}_\mathrm{r}} -\frac{q}{2}\|\Delta t_\mathrm{r}\|^2 + (\lambda^*)^T V_\mathrm{r}\Delta t_\mathrm{r}\,.$$

[3]The maximum ratios between the largest and the smallest eigenvalue of the Hessian (*condition numbers*) are between $2 \cdot 10^3$ and $5 \cdot 10^3$ depending on the problem size and the scenario.

In the following section we will investigate the resource-efficient solution of the dual problem (8) using gradient methods. For more details on the used dualization technique, which is also referred to as *partial elimination*, we refer the reader to, e.g., [21, §4.2.2].

III. Gradient Methods for MP^{3}C

This section first provides in Section III-A a condensed introduction to the gradient methods employed for MP^{3}C. In particular, it highlights the main issues with respect to our application, i.e. assumptions for convergence, convergence speed and effects in case of inexact computations. In Section III-B we then provide all the necessary details so that the discussed gradient methods can be applied to efficiently solve the dual MP^{3}C problem in (8).

A. A Primer on Gradient Methods

The MP^{3}C problem belongs to the class of convex optimization problems in which a convex function is minimized over a convex set, see, e.g., [22]. There is a great variety of different solution methods for convex problems. It is important to understand that each of them has its own strengths and weaknesses with respect to the additional assumptions made (level of smoothness, type of feasible set, etc.), the computational cost per iteration (expensive solution of system of linear equations versus cheap matrix-vector products), the numerical stability in the presence of inaccurate computations and other criteria.

For the MP^{3}C problem, three main solution methods come into question: interior point, active set and gradient methods. Interior point methods (cf. [23, §16.7]) and active set methods (cf. [23, §16.4]) are traditionally used as general purpose methods. Fast specializations of them, e.g. for MPC in [24], [25], are primarily obtained by exploiting structure in the problem data to speed up linear algebra operations. For MP^{3}C, these are not the methods of choice as hardware resource usage on platforms in mind is too high (some parts of the algorithms still require floating-point arithmetic for numerical stability) and costly divisions cannot be avoided.

Gradient methods find an optimizer by iteratively evaluating function values and gradients at test points and updating the solution estimate. They are resource-efficient by their very nature: Function and gradient evaluations are cheap in numerous important cases, and for many problems of practical interest, such as MP^{3}C, they can be made divisionfree (and free of other more sophisticated mathematical functions). However, these methods are not considered general purpose methods since their convergence rate can vary widely depending on the problem characteristic. Since many practical applications only require solutions with low accuracy, e.g. within 1 % of the optimal value, gradient methods can still be a viable alternative, particularly if problem specific properties, such as the geometry of the feasible set, are fully exploited.

From here on we investigate both the classic and Nesterov's fast gradient method for the solution of the dual MP^{3}C problem. We first review their associated algorithms, given

Algorithm 1 Classic Gradient Method for Problem (10)

```
Require: Initial iterate $x_0 \in \mathbb{R}^{n_f}$, factor $h \in (0,2)$
1: loop
2:     $x_{i+1} = x_i - \frac{h}{L}\nabla f(x_i)$
3: end loop
```

Algorithm 2 Fast Gradient Method for Problem (10)

```
Require: Initial iterate $x_0 \in \mathbb{R}^{n_f}$, $y_0 = x_0$,
    initial weight $\alpha_0 : \sqrt{\mu/L} \le \alpha_0 < 1$
1: loop
2:     $x_{i+1} = y_i - \frac{1}{L}\nabla f(y_i)$
3:     $\alpha_{i+1} \in (0,1) : \alpha_{i+1}^2 = (1-\alpha_{i+1})\alpha_i^2 + (\mu/L)\alpha_{i+1}$
4:     $\beta_i = \frac{\alpha_i(1-\alpha_i)}{\alpha_i^2+\alpha_{i+1}}$
5:     $y_{i+1} = x_{i+1} + \beta_i(x_{i+1} - x_i)$
6: end loop
```

in Algorithms 1 and 2, and their assumptions for convergence by means of the unconstrained optimization problem

$$\min_x f(x) \tag{10}$$

with convex objective function $f : \mathbb{R}^{n_f} \to \mathbb{R}$.

Prerequisite for the classic and the fast gradient method to converge is a Lipschitz continuous gradient ∇f, i.e. existence of a constant $L > 0$ such that for all $x, y \in \mathbb{R}^{n_f}$

$$f(x) - f(y) - \nabla f(y)^T(x-y) \le \frac{L}{2}\|x-y\|^2 \,. \tag{11}$$

It can be shown [14, §2] that in this so-called *smooth* case, the classic gradient method requires $\mathcal{O}(L/\epsilon)$ iterations to find an approximate solution that is ϵ-close to the optimal value. For the fast gradient method, the number of iterations only grows with $\mathcal{O}(\sqrt{L/\epsilon})$. In both cases, convergence is *sublinear*.

If function f is also *strongly convex*, i.e. there is a constant $\mu > 0$ such that for all $x, y \in \mathbb{R}^{n_f}$ we have

$$\frac{\mu}{2}\|x-y\|^2 \le f(x) - f(y) - \nabla f(y)^T(x-y)\,, \tag{12}$$

the number of iterations grows with $\mathcal{O}(\kappa_f \ln(L/\epsilon))$ for the classic gradient method and $\mathcal{O}(\sqrt{\kappa_f}\ln(L/\epsilon))$ for the fast gradient method ($\kappa_f = L/\mu$ is the *condition number* of f). This is denoted *linear* convergence.

Implementations in fixed-point arithmetic inherently suffer from round-off errors. These errors add up with errors from inexact gradient evaluations, e.g. when the gradient is computed by means of an iterative solution to a subproblem (this is indeed the case for the dual formulation of the MP^{3}C problem as shown below). However, the classic gradient method can tolerate such errors without diverging. This is true for the smooth [26] and the strongly convex case [27]. For the fast gradient method the situation is different. Accelerated convergence comes at the price of error accumulation which is unbounded in the smooth case [26] but bounded in the strongly convex case [27].

B. Application to MP^{3}C

For the dual MP^{3}C problem in (8), the gradient of the dual function (9) is given by

$$\nabla d(\lambda) = \lambda + \psi_{\text{s,err}} + V_{\text{r}}\Delta t_{\text{r}}^*(\lambda) \tag{13}$$

where

$$\Delta t_{\mathrm{r}}^*(\lambda) = \operatorname*{arg\,max}_{\Delta t_{\mathrm{r}} \in \mathbb{X}_{\mathrm{r}} - \bar{t}_{\mathrm{r}}} -\frac{q}{2}\|\Delta t_{\mathrm{r}}\|^2 + \lambda^T V_{\mathrm{r}} \Delta t_{\mathrm{r}} \,. \tag{14}$$

Note that the gradient of the max-term in (8) is due to Danskin's Theorem [21, Proposition B.25].

It can be seen that the latter subproblem can be rewritten as a projection problem

$$\Delta t_{\mathrm{r}}^*(\lambda) = \operatorname*{arg\,min}_{\Delta t_{\mathrm{r}} \in \mathbb{X}_{\mathrm{r}} - \bar{t}_{\mathrm{r}}} \frac{1}{2}\|\Delta t_{\mathrm{r}} - q^{-1} V_{\mathrm{r}}^T \lambda\|^2 \,. \tag{15}$$

For notational convenience, we denote the projection of the point z on the set $\mathbb{Q}$ as $\pi_{\mathbb{Q}}(z)$ from here on so that

$$\Delta t_{\mathrm{r}}^*(\lambda) = \pi_{\mathbb{X}_{\mathrm{r}} - \bar{t}_{\mathrm{r}}}\left(q^{-1} V_{\mathrm{r}}^T \lambda\right) \,,$$

which can further be simplified to

$$\Delta t_{\mathrm{r}}^*(\lambda) = \pi_{\mathbb{X}_{\mathrm{r}}}\left(q^{-1} V_{\mathrm{r}}^T \lambda + \bar{t}_{\mathrm{r}}\right) - \bar{t}_{\mathrm{r}} \tag{16}$$

by means of a variable transformation in (15). The latter form will be the starting point for the investigations in Section III-B1 where an efficient method for the evaluation of the projection operator $\pi_{\mathbb{X}_{\mathrm{r}}}(\cdot)$ in fixed-point arithmetic will be derived.

Before looking at the projection operation more closely, it remains to calculate constants L_d (Lipschitz constant of dual gradient) and μ_d (strong convexity parameter of dual objective). These constants determine the step sizes h/L_d in the classic gradient method as well as $1/L_d$ and β_i in the fast gradient method. Since the dual function (9) is the sum of a quadratic function with identity Hessian and a max-term, constant L_d itself is given by the sum

$$L_d = 1 + q^{-1} \lambda_{\max}\left(V_{\mathrm{r}} V_{\mathrm{r}}^T\right) \,, \tag{17}$$

where the second term is the tight Lipschitz constant of the max-term's gradient as proved in [28]. Here, $\lambda_{\max}(\cdot)$ denotes the largest eigenvalue. Indeed, the sum (17) is the *tight* Lipschitz constant of the dual's gradient.

The Lipschitz constant is a function of matrix V_{r}, which is a parameter in MP^{3}C. Note that the positive definite 2×2-matrix $V_{\mathrm{r}} V_{\mathrm{r}}^T$ can be written as

$$V_{\mathrm{r}} V_{\mathrm{r}}^T = \left(\frac{V_{\mathrm{dc}}}{6}\right)^2 \begin{bmatrix} 4n_{\mathrm{a}} + n_{\mathrm{b}} + n_{\mathrm{c}} & \sqrt{3}\left(n_{\mathrm{c}} - n_{\mathrm{b}}\right) \\ \sqrt{3}\left(n_{\mathrm{c}} - n_{\mathrm{b}}\right) & 3\left(n_{\mathrm{b}} + n_{\mathrm{c}}\right) \end{bmatrix} \tag{18}$$

since, e.g. for phase 'a', $\|\Delta u_{\mathrm{a}}\|^2 = n_{\mathrm{a}}$ by definition of Δu_{a}. Hence, only the number of transitions in the respective phases are important for the maximum eigenvalue; Transition *directions* do not play any role.

In the 2×2 case we can write down the maximum eigenvalue explicitly so that the dual Lipschitz constant follows as

$$L_d = 1 + q^{-1} \frac{V_{\mathrm{dc}}^2}{18} \cdot \left(n_{\mathrm{a}} + n_{\mathrm{b}} + n_{\mathrm{c}} + \sqrt{n_{\mathrm{a}}^2 + n_{\mathrm{b}}^2 + n_{\mathrm{c}}^2 - n_{\mathrm{a}} n_{\mathrm{b}} - n_{\mathrm{a}} n_{\mathrm{c}} - n_{\mathrm{b}} n_{\mathrm{c}}}\right). \tag{19}$$

Online computation of this constant requires squares and square roots to be evaluated and thus is not a valid option if we aim for a ultralow-resource implementation. As an alternative, all possible Lipschitz constants (or preferrably, step sizes h/L_d and $1/L_d$) can be enumerated and stored in a lookup table. A naive implementation requires n^3 entries. If we exploit symmetry of the Lipschitz constant L_d in the arguments $n_{\mathrm{a}}, n_{\mathrm{b}}$ and n_{c}, the number of entries in the lookup table can be reduced to $n^2 + n!/\left(3!\,(n-3)!\right)$ and thus a resource-saving implementation can be achieved.

The parameter of strong convexity is solely determined by the quadratic term in the dual function, i.e.

$$\mu_d = 1 \tag{20}$$

and thus is not dependent on any problem data. Note that the max-term does not permit any quadratic lower bound (12). In order to see this, it suffices to consider the simple case $\bar{t}_{\mathrm{r}} = 0$ and a line segment $[\lambda_1, \lambda_2] \subseteq \mathbb{R}^2$ in the dual domain such that the projection $\Delta t_{\mathrm{r}}^*(\lambda)$ in (16) evaluates to the origin for all points λ in the line segment. In this case, the max-term in the dual function (9) is an affine function over the line segment and therefore, no quadratic lower bound can exist.

Finally, we note that for the fast gradient method the step size β_i for the update of the secondary iterate y_{i+1} depends on both the iteration counter $i \in \{1, \ldots, i_{\max}\}$ *and* the inverse condition number μ_d/L_d (cf. line 3 in Algorithm 2). For an implementation that is gentle on resource usage, saving all step size sequences $\{\beta_i\}_{i=1}^{i_{\max}}$ for every possible inverse condition number is not acceptable. As an alternative, we opt for saving this sequence only *once* for the smallest inverse condition number $1/(1 + q^{-1} n V_{\mathrm{dc}}^2/6)$ which is the inverse of the largest Lipschitz constant[4]. This choice still guarantees linear convergence since we effectively underestimate the strong convexity parameter μ_d. However, every underestimate of the tight strong convexity parameter in (20) fulfills the inequality in (12) and hence is also an admissible strong convexity parameter.

1) Efficient Projection On the Truncated Monotone Cone: In the dual reformulation of the MP^{3}C problem the projection operator $\pi_{\mathbb{X}_{\mathrm{r}}}(\cdot)$ has to be evaluated once in every iteration of the gradient methods in order to compute the dual gradient (cf. (13) and (16)). This section will discuss all necessary details in order to make this projection operation resource-efficient. In the following we will first decompose the problem according to each phase. The feasible set for each phase is a truncated monotone cone (cf. (6)), which is the intersection of a monotone cone and a box. We will present a new result that says that the projection on this set is equivalent to projecting *first* on the monotone cone and *then* on the box. This result will be key to the suggested projection implementations thereafter.

Since set $\mathbb{X}_{\mathrm{r}}$ is the direct product of sets $\mathbb{X}_{\mathrm{r,a}}, \mathbb{X}_{\mathrm{r,b}}$ and $\mathbb{X}_{\mathrm{r,c}}$, the projection operation in (16) can be decomposed into three independent projections. In fact, the point to project of length $3n$ is split into three chunks of size n that are individually projected on the associated sets $\mathbb{X}_{\mathrm{r,a}}, \mathbb{X}_{\mathrm{r,b}}$ and $\mathbb{X}_{\mathrm{r,c}}$. For that reason it suffices to consider projection on a single truncated monotone cone from here on. Since the following results are generally valid, we use a different, more standard notation as in the MP^{3}C context.

[4]The largest Lipschitz constant is obtained if (19) is maximized over all triples $\left(n_{\mathrm{a}}, n_{\mathrm{b}}, n_{\mathrm{c}}\right) \in \{1, \ldots, n\}^3$. The Lipschitz constant can be shown to be a nondecreasing function of $\left(n_{\mathrm{a}}, n_{\mathrm{b}}, n_{\mathrm{c}}\right)$, hence, the maximum is attained at (n, n, n).

We consider the truncated monotone cone of dimension m

$$\overline{\mathbb{K}} = \left\{ x \in \mathbb{R}^m \,|\, \underline{x} \leq x_1 \leq x_2 \leq \ldots \leq x_m \leq \overline{x} \right\}, \tag{21}$$

which can be written as the intersection of the convex monotone cone $\mathbb{K}$ and a box $\mathbb{B}$, i.e.

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

where

$$\mathbb{K} = \left\{ x \in \mathbb{R}^m \,|\, x_1 \leq x_2 \leq \ldots \leq x_m \right\}, \tag{22a}$$
$$\mathbb{B} = \left\{ x \in \mathbb{R}^m \,|\, \underline{x} \leq x_i \leq \overline{x},\, i \in \{1, 2, \ldots, m\} \right\}. \tag{22b}$$

It turns out that projection on set $\overline{\mathbb{K}}$ can be further decomposed. In particular, we will prove in Theorem 1 below that $\pi_{\overline{\mathbb{K}}}(z) = \pi_{\mathbb{B}}(\pi_{\mathbb{K}}(z))$. In the proof, the following proposition and corollary are central.

Proposition 1 (Optimality in Conic Convex Optimization)**.** *Consider the convex optimization problem*

$$\min_{x \in \mathbb{C}} f(x)$$

where $f : \mathbb{R}^m \to (-\infty, \infty]$ is a closed proper convex function and $\mathbb{C} \subseteq \mathbb{R}^m$ is a closed convex cone. If the optimal value is finite and there exists a point in the intersection of the relative interiors of $\mathbb{C}$ and $\operatorname{dom} f$, then (x^*, s^*) is a primal/dual solution pair if and only if

$$x^* \in \arg\min_x f(x) - (s^*)^T x \tag{i}$$
$$x^* \in \mathbb{C} \tag{ii}$$
$$s^* \in \mathbb{C}^* \tag{iii}$$
$$(s^*)^T x^* = 0, \tag{iv}$$

where $\mathbb{C}^* = \left\{ s \in \mathbb{R}^m \,|\, s^T y \geq 0 \text{ for all } y \in \mathbb{C} \right\}$ denotes the dual cone of $\mathbb{C}$.

Proof. This follows from [29, Prop. 5.3.8] with $f_1 = f$, f_2 being the indicator function of $\mathbb{C}$, $A = I_m$ and noticing that

$$x^* \in \arg\min_{y \in \mathbb{C}} (s^*)^T y \Leftrightarrow x^* \in \mathbb{C},\ s^* \in \mathbb{C}^* \text{ and } (s^*)^T x^* = 0. \quad \square$$

Corollary 1 (Projection on Convex Cone)**.** *Point $\pi_{\mathbb{C}}(z)$ is the projection of z on the nonempty convex cone $\mathbb{C}$ if and only if*

$$\pi_{\mathbb{C}}(z) \in \mathbb{C},\ \ \pi_{\mathbb{C}}(z) - z \in \mathbb{C}^* \text{ and } (\pi_{\mathbb{C}}(z) - z)^T \pi_{\mathbb{C}}(z) = 0.$$

Proof. We let $f(x) = \frac{1}{2}\|x - z\|^2$ in Proposition 1 and notice that all assumptions are satisfied. Then condition (i) can be rewritten as $\pi_{\mathbb{C}}(z) = z + s^*$ and the result follows. $\square$

We now state the main theoretical contribution of this paper.

Theorem 1 (Projection on Truncated Monotone Cone)**.** *Consider the nonempty truncated monotone cone $\overline{\mathbb{K}}$ in* (21). *For the projection of point $z \in \mathbb{R}^m$ on this set it holds that*

$$\pi_{\overline{\mathbb{K}}}(z) = \pi_{\mathbb{B}}(\pi_{\mathbb{K}}(z)),$$

where $\mathbb{K}$ is the monotone cone in (22a) *and $\mathbb{B}$ ist the m-dimensional box in* (22b)*.*

Proof. The problem of projection on set $\overline{\mathbb{K}}$ is recovered if in Proposition 1 we let $\mathbb{C} = \mathbb{K}$ and $f(x) = \frac{1}{2}\|x - z\|^2 + \iota_{\mathbb{B}}(x)$, where $\iota_{\mathbb{B}}(\cdot)$ denotes the indicator function for set $\mathbb{B}$. We will show that the optimality conditions (i)-(iv) are satisified with $x^* = \pi_{\mathbb{B}}(\pi_{\mathbb{K}}(z))$ and $s^* = \pi_{\mathbb{K}}(z) - z$.
In order to validate (i), it suffices to rewrite it as a projection,

$$x^* = \pi_{\mathbb{B}}(z + s^*),$$

which is clearly satisfied by our choices for x^* and s^*. Constraint (ii) is fulfilled since projection of an ordered vector on a box results in an ordered vector again, hence $\pi_{\mathbb{B}}(\pi_{\mathbb{K}}(z)) \in \mathbb{K}$. We make use of Corollary 1 to confirm that $\pi_{\mathbb{K}}(z) - z \in \mathbb{K}^*$ which validates (iii). Last, we prove that (iv) holds by

$$\begin{aligned} 0 &\leq \big(\pi_{\mathbb{K}}(z) - z\big)^T \pi_{\mathbb{B}}(\pi_{\mathbb{K}}(z)) \\ &= \big(\pi_{\mathbb{K}}(z) - z\big)^T \pi_{\mathbb{K}}(z) - \big(\pi_{\mathbb{K}}(z) - z\big)^T \big(\pi_{\mathbb{K}}(z) - \pi_{\mathbb{B}}(\pi_{\mathbb{K}}(z))\big) \\ &\leq \big(\pi_{\mathbb{K}}(z) - z\big)^T \pi_{\mathbb{K}}(z) = 0. \end{aligned}$$

The first inequality follows from the definition of the dual cone and the fact that the first vector of the scalar product is in $\mathbb{K}^*$ whereas the other vector is in $\mathbb{K}$. Also we note that

$$\pi_{\mathbb{K}}(z) - \pi_{\mathbb{B}}(\pi_{\mathbb{K}}(z)) = \begin{bmatrix} \big(\pi_{\mathbb{K}}(z)\big)_{\underline{I}} \ - \ \underline{x} \cdot 1_{|\underline{I}|} \\ 0_{m - |\underline{I}| - |\overline{I}|} \\ \big(\pi_{\mathbb{K}}(z)\big)_{\overline{I}} \ - \ \overline{x} \cdot 1_{|\overline{I}|} \end{bmatrix}, \tag{24}$$

where index sets $\underline{I}$ and $\overline{I}$ contain indices of active box constraints of $\pi_{\mathbb{B}}(\pi_{\mathbb{K}}(z))$, i.e.

$$\underline{I} = \left\{ i \in \{1, 2, \ldots, m\} \,|\, \big(\pi_{\mathbb{B}}(\pi_{\mathbb{K}}(z))\big)_i = \underline{x} \right\}$$
$$\overline{I} = \left\{ i \in \{1, 2, \ldots, m\} \,|\, \big(\pi_{\mathbb{B}}(\pi_{\mathbb{K}}(z))\big)_i = \overline{x} \right\}.$$

Since the vector in (24) is included in cone $\mathbb{K}$, the second inequality follows by similar reasoning as for the first inequality. The final equality follows from Corollary 1. $\square$

Remark 1. *Theorem 1 is a generalization of [30, Theorem 1] where projection on the nonnegative monotone cone is proved, i.e. $\underline{x} = 0$, $\overline{x} = \infty$. Note that the line of proof pursued here is different from the one presented in [30].*

In [13] the authors have noticed empirically the validity of Theorem 1. However, no proof is given there.

The main benefit of Theorem 1 in the context of MP^{3}C is that the projection on the truncated monotone cone can be made a two-stage process: Projection on the monotone cone followed by a projection on the box. The former operation cannot be done in closed form but iterative algorithms exist that provide the exact projection in at most m iterations (cf. [31]). The latter operation can be implemented cheaply in hardware by comparison and saturation only, i.e.

$$\big(\pi_{\mathbb{B}}(z)\big)_i = \max\big(\underline{x}, \min(\overline{x}, z_i)\big), \quad i \in \{1, 2, \ldots, m\}.$$

In the remainder of this section we will emphasize on a resource-efficient implementation of the projection on the monotone cone $\pi_{\mathbb{K}}(\cdot)$. Unfortunately, the algorithm presented in [31] is not well suited for this purpose as it involves pseudoinverses of matrices of varying size. A more suitable approach is to precompute the projection operator by means of multiparametric programming with the point to project $z \in \mathbb{R}^m$ being the parameter (this approach was taken in [13]). The explicit solution is a polyhedral partition of the parameter space $\mathbb{R}^m$ and an online lookup suffices to evaluate the projection operation. For this approach, Theorem 1 leads to a significantly more resource-efficient lookup table. Whereas in MP^{3}C the explicit solution for the projection on the truncated monotone cone possesses $2^{n+1}-1$ polyhedral regions in the parameter space of dimension $n+1$ (the upper bound is an additional parameter there), we now need to store only 2^{n-1} regions in n dimensions for the projection on the monotone cone. (These numbers follow from computations in MPT3 [17].) Although the number of regions grows exponentially with n in both cases, Theorem 1 leads to a reduction of almost 75%.

However, the explicit solution has drawbacks. It needs a lookup table and an efficient method to solve the point location problem, which requires at least one DSP-type multiplier. We propose to solve the projection onto the monotone cone *approximately in the dual domain*. Most importantly, this approach does not require any DSP-type multiplier. In order to see this, note that the solution to the projection problem can be computed from the dual solution $\eta^* \in \mathbb{R}^{m-1}$ as

$$\pi_{\mathbb{K}}(z) = z - G^T\eta^* \,, \tag{25a}$$

$$\eta^* = \arg\min_{\eta\geq 0} \frac{1}{2}\eta^T GG^T\eta - (Gz)^T\eta \,. \tag{25b}$$

Herein, G denotes the $(m-1)\times m$-matrix

$$\begin{bmatrix} 1 & -1 & 0 & \cdots & 0 \\ 0 & 1 & -1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 & -1 \end{bmatrix}, \tag{26}$$

which defines the monotone cone by $\mathbb{K} = \{x \in \mathbb{R}^m | Gx \leq 0\}$. For the solution of the dual problem (25b) we employ the classic gradient method in Algorithm 1 with step size $h/L_\eta = 2/(L_\eta + \mu_\eta)$ which is a valid choice in the strongly convex case (cf. [32, Theorem 5.5]). Since the dual projection problem is a constrained problem, every gradient step requires a projection onto the feasible set, which is the LP-cone here (cf. [14, §2.2.3]). The algorithmic scheme for the solution of the dual projection problem on the monotone cone (25b) can thus be written as

$$\eta_{i+1} = \max\Big(0_{m-1},\, \eta_i - \frac{2}{L_\eta + \mu_\eta}\big(GG^T\eta_i - Gz\big)\Big). \tag{27}$$

Since matrix G is a first difference matrix, product Gz can be computed without DSP-type multipliers. Furthermore, Hessian GG^T is a second difference matrix, e.g. for $m = 4$

$$GG^T = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}. \tag{28}$$

Conclusively, only shifts, negations and additions are required for multiplication with this matrix. Ultimately, multiplying in the step size can be implemented by a shift operation too as follows from the next proposition.

Proposition 2. *Consider the dual projection problem for the monotone cone $\mathbb{K}$ given in* (25b). *Let L_η denote the Lipschitz constant of the gradient and μ_η the strong convexity parameter of the objective. For every dimension m of the cone $\mathbb{K}$ the step size for the classic gradient method in* (27) *is given by*

$$\frac{2}{L_\eta + \mu_\eta} = \frac{1}{2}\,.$$

Proof. The Lipschitz constant of the gradient of a convex quadratic objective is given by the maximum eigenvalue of the Hessian, the strong convexity parameter by the minimum eigenvalue (cf. [14, Theorems 2.1.6 and 2.1.11]). For a second difference matrix, these eigenvalues are

$$\lambda_{\max}\left(GG^T\right) = 2 - 2\cos\frac{(m-1)\pi}{m}\,,$$

$$\lambda_{\min}\left(GG^T\right) = 2 - 2\cos\frac{\pi}{m}\,,$$

as follows from the theory of eigenvalues of Toeplitz Matrices. Since $2\cos\frac{(m-1)\pi}{m} = -2\cos\frac{\pi}{m}$ the result follows. □

This result allows us to implement the projection on the monotone cone (and hence the truncated monotone cone) without any DSP-type multipliers. Most importantly, preliminary computational studies have shown that when employing warmstarting from the last iterate η_{i+1} of the previous outer iteration, a *single iteration* of the dual projection method in (27) suffices for computing an approximate gradient from (13) so that both the classic and the fast gradient method converge. The reason for this is the very small condition number $\lambda_{\max}\left(GG^T\right)/\lambda_{\min}\left(GG^T\right) < 10$ of the Hessian in (25b) due to small values of m in MP^{3}C, i.e. $m = n \in \{3,4,5\}$.

2) Preconditioning: The dual MP^{3}C problem (8) is a strongly convex optimization problem. As discussed in Section III-A, both the classic and the fast gradient method converge linearly in this case with the number of iterations to reach an ϵ-suboptimal solution being proportional to the condition number $\kappa_d = L_d/\mu_d$ (classic gradient method) or its square root $\sqrt{\kappa_d}$ (fast gradient method). The condition number of the dual problem can be influenced by a change of variables $\lambda = P\gamma$ with P being an invertible 2×2-matrix. Using a recent result from [33] it can be shown that the condition number of the dual objective (9) under a change of variables is given by

$$\kappa_d(P,j) = \frac{\lambda_{\max}\left(P\big(I + q^{-1}M_j\big)P^T\right)}{\lambda_{\min}\left(PP^T\right)}\,, \tag{29}$$

where matrix M_j denotes the jth realization of positive definite matrix $V_r{V_r}^T$ in (18). It can be seen that there exists a total of n^3 possible realizations of this matrix depending on the number of transitions per phase $(n_{\mathrm{a}}, n_{\mathrm{b}}, n_{\mathrm{c}})$. For every realization M_j we could find a specific preconditioner matrix P_j that minimizes the condition number $\kappa_d(j)$ from the solution to a convex semidefinite program (cf. [34, §3.2]). However, this approach requires n^3 2×2-matrices to be stored which is not a viable approach on the FPGA in mind due to resource constraints. We therefore aim for a single, static preconditioner P^* that minimizes the worst case condition number of the dual objective, i.e.

$$P^* \in \arg\min_{P \text{ invertible}} \max_{j\in\{1,2,\ldots,n^3\}} \kappa_d(P,j)\,. \tag{30}$$

It turns out that there is no reason for choosing a preconditioner different than the identity matrix in MP^{3}C, i.e. static preconditioning does not pay off in this application. The following proposition establishes this result.

Proposition 3. *Consider the computation of a static preconditioner P^* according to the min-max problem in* (30) *with condition number $\kappa_d(P,j)$ defined in* (29)*. A solution to this problem is given by the identity matrix.*

Proof. Choose j such that M_j is the realization of matrix $V_r V_r^T$ for the case $n_\text{a} = n_\text{b} = n_\text{c} = n$. According to (18), $M_j = \nu I$ with $\nu = nV_\text{dc}^2/6$ in this case. Hence,

$$\begin{aligned}\min_{P \text{ inv.}} \max_{j\in\{1,\dots,n^3\}} \kappa_d(P,j) &\geq \min_{P \text{ inv.}} \frac{\lambda_{\max}\left(P\big(I+q^{-1}\nu I\big)P^T\right)}{\lambda_{\min}\left(PP^T\right)}\\ &= (1+q^{-1}\nu)\min_{P \text{ inv.}} \frac{\lambda_{\max}\left(PP^T\right)}{\lambda_{\min}\left(PP^T\right)}\\ &= 1+q^{-1}\nu\,.\end{aligned}$$

On the other hand, choose $P = I$ such that

$$\begin{aligned}\min_{P \text{ inv.}} \max_{j\in\{1,\dots,n^3\}} \kappa_d(P,j) &\leq \max_{j\in\{1,\dots,n^3\}} \kappa_d(I,j)\\ &= 1+q^{-1}\cdot \max_{j\in\{1,\dots,n^3\}} \lambda_{\max}\left(M_j\right)\,,\end{aligned}$$

where $\max_{j\in\{1,\dots,n^3\}} \lambda_{\max}\left(M_j\right) = \nu$ as follows from Section III-B. This proves that the identity is a solution to the min-max problem. □

3) Choice of Step Size Enlargement Factor: In the classic gradient method in Algorithm 1 the step size is given by h/L where L denotes the Lipschitz constant of the objective's gradient. The gradient method can be proved to converge for any $h \in (0,2)$ where $h = 1$ is recommended in the literature. Very recent research in [35] suggests that for better convergence speed the standard step size of $1/L$ should be enlarged for most purposes, i.e. a *step size enlargement factor* $h > 1$ should be chosen. The optimal choice for h is shown to be dependent on three factors: the performance criterion, the condition number and the conducted number of iterations.

Since the MP^{3}C problem is solved in the dual domain, the 'residual gradient norm' [35, §4.1.3] is the performance criterion of choice. Based on the derivations in the cited work we have chosen a static, optimal step size enlargement factor for an *a priori* fixed number of iterations and the worst case condition number $1+q^{-1}nV_\text{dc}^2/6$ in this application.

IV. FPGA Implementation Aspects

In this section we focus on implementation aspects of the gradient methods for MP^{3}C that were described in Section III. The most important aspect with respect to a fixed-point implementation on an FPGA is the prevention of overflows. Section IV-A presents an analysis that allows one to derive an upper bound on the largest absolute number that can occur inside the gradient methods. Knowledge of this number permits one to safely pick the number of integer bits for a fixed-point number representation. In the dual form of the MP^{3}C problem there are both dual and primal variables (the latter are used to compute the dual gradient). Section IV-B introduces an appropriate scaling for the dual iterates of the gradient methods so that the number of fractional bits are primarily determined by the accuracy requirement in the primal domain. As a side effect, the number of DSP-type multipliers is reduced, too. Finally, Section IV-C provides an estimate of the number of clock cycles for a single iteration of the employed gradient methods.

A. Prevention of Arithmetic Overflow

In order to compute an upper bound on the largest absolute number inside the gradient methods, we first define appropriate parameter sets for the continuous parameters of the MP^{3}C problem. These parameters are contained in the following compact sets

$$\psi_\text{s,err} \in [-\overline{\psi_\text{s,err}}, \overline{\psi_\text{s,err}}]\,, \tag{31a}$$

$$\bar{t}_\text{r} \in \mathbb{X}_\text{r}\,, \tag{31b}$$

$$\bar{t}_{\text{a}_{n_\text{a}+1}}, \bar{t}_{\text{b}_{n_\text{b}+1}}, \bar{t}_{\text{c}_{n_\text{c}+1}} \in [0, \bar{t}_\text{max}]\,. \tag{31c}$$

For the sake of brevity, we only consider the case of the classic gradient method. The classic gradient method can be summarized from Section III in a single line as the iteration

$$\lambda_{i+1} = \lambda_i - \frac{h}{L_d}\left(\lambda_i + \psi_\text{s,err} + V_\text{r}\left(\pi_{\mathbb{X}_\text{r}}\left(q^{-1}V_\text{r}^T\lambda_i + \bar{t}_\text{r}\right) - \bar{t}_\text{r}\right)\right), \tag{32}$$

where the step size h/L_d is a function of V_r according to (17) and the projection $\pi_{\mathbb{X}_\text{r}}(\cdot)$ is evaluated approximately via (27) and by making use of Theorem 1.

First and foremost we derive an upper bound on $\|\lambda_i\|_\infty$ which is valid for all $i \geq 0$ and every parameter scenario. For the classic gradient method we have for all $i \geq 0$ [14, §2.1.5]

$$\|\lambda_{i+1} - \lambda^*\| \leq \|\lambda_i - \lambda^*\|\,.$$

In MP^{3}C we choose $\lambda_0 = 0$ as an initial iterate, therefore

$$\|\lambda_i - \lambda^*\| \leq \|\lambda^*\| \quad \text{for all } i \geq 0\,.$$

Making use of the triangular inequality gives the upper bound

$$\|\lambda_i\|_\infty \leq \|\lambda_i\| \leq 2\|\lambda^*\| \quad \text{for all } i \geq 0\,.$$

So in order to bound the dual iterates, we need to bound the Lagrange multiplier λ^*. This can be accomplished as follows. From the necessary and sufficient optimality condition for the dual problem, $\nabla d(\lambda^*) = 0$, we conclude that

$$\|\lambda^*\| = \|\psi_\text{s,err} + V_\text{r}\Delta t_\text{r}^*(\lambda^*)\|$$

using the definition of the dual gradient in (13). Since the optimal primal solution Δt_r^* is identical with $\Delta t_\text{r}^*(\lambda^*)$, we obtain the chain of inequalities

$$\|\psi_\text{s,err} + V_\text{r}\Delta t_\text{r}^*\|^2 \leq \|\psi_\text{s,err} + V_\text{r}\Delta t_\text{r}^*\|^2 + q\|\Delta t_\text{r}^*\|^2 \leq \|\psi_\text{s,err}\|^2.$$

The middle term is twice the minimum value of (5) and the rightmost term is twice the primal objective value if $\Delta t_\text{r} = 0$, which is feasible in view of constraint (31b). Consequently, we obtain

$$\|\lambda_i\|_\infty \leq 2\|\overline{\psi_\text{s,err}}\| \quad \text{for all } i \geq 0\,. \tag{33}$$

We also need to compute an upper bound on the variables in the approximate evaluation of the projection on the monotone cone through iteration (27) followed by (25a). As our computational studies have shown, only a single iteration suffices for both the classic and the fast gradient method to converge if warmstarting is employed. Warmstarting is beneficial for convergence but more difficult to analyze than coldstarting before. However, by introducing conservatism appropriately, a reasonable upper bound on the first iterate can be derived as shown next.

From here on we denote by $\eta_{0(i)}$ the initial iterate for iteration (27) at outer iteration i. Similarly, $\eta_{1(i)}$ denotes the first update and $\eta^*_{(i)}$ the optimal solution at outer iteration i. We let $\eta_{0(1)} = 0$ and deduce from [32, Theorem 5.5]

$$\|\eta_{1(1)} - \eta^*_{(1)}\| \leq c\,\|\eta^*_{(1)}\|\,, \tag{34}$$

where convergence ratio $c = (\kappa_\eta - 1)/(\kappa_\eta + 1)$ and $\kappa_\eta = L_\eta/\mu_\eta$ is the condition number of the objective in (25b). For the second outer iteration we similarly obtain

$$\|\eta_{1(2)} - \eta^*_{(2)}\| \leq c\,\|\eta_{0(2)} - \eta^*_{(2)}\|\,.$$

Considering warmstarting, i.e. $\eta_{0(2)} = \eta_{1(1)}$, and (34) gives

$$\|\eta_{1(2)} - \eta^*_{(2)}\| \leq c^2\|\eta^*_{(1)}\| + c\|\eta^*_{(1)} - \eta^*_{(2)}\|\,.$$

If we carry on this argument and use the triangular inequality, we arrive at the following upper bound for the size of the first iterate after outer iteration $i \geq 2$

$$\|\eta_{1(i)}\| \leq \|\eta^*_{(i)}\| + c^i\|\eta^*_{(1)}\| + \sum_{j=1}^{i-1} c^j\,\|\eta^*_{(i-j)} - \eta^*_{(i-j+1)}\|\,.$$

Let ζ denote a bound (to be derived below) on the largest dual optimal solution, i.e. $\|\eta^*_{(i)}\| \leq \zeta$ for all $i \geq 0$. With this bound, the latter inequality becomes

$$\|\eta_{1(i)}\| \leq \zeta\big(1 + c^i + 2\sum_{j=1}^{i-1} c^j\big) \leq \zeta\frac{1+c}{1-c} \quad \text{for all } i \geq 0\,.$$

Plugging in the definition of the convergence ratio c and considering that the condition number of the dual objective in (25b) for projection on a monotone cone in dimension n can be compactly written as $\kappa_\eta = \cot^2\!\big(\frac{\pi}{2n}\big)$ results in the following upper bound for the first iterate

$$\|\eta_{1(i)}\|_\infty \leq \zeta\cot^2\!\left(\frac{\pi}{2n}\right) \quad \text{for all } i \geq 0\,.$$

We are left to derive an upper bound ζ on the norm of the largest dual optimal solution. For this, we assume that z_i is the point to project onto the monotone cone at outer iteration i. From the optimality condition in Corollary 1 it follows using (25a) and the Cauchy-Schwarz inequality

$$\|G^T\eta^*_{(i)}\|^2 = \big(G^T\eta^*_{(i)}\big)^T z_i \leq \|G^T\eta^*_{(i)}\|\,\|z_i\|\,,$$

so that $\|G^T\eta^*_{(i)}\| \leq \|z_i\|$ if $\eta^*_{(i)} \neq 0$. The minimum singular value of matrix G^T is given by $\sqrt{2-2\cos\frac{\pi}{n}}$ (cf. proof of Proposition 2) so that an upper bound on $\|\eta^*_{(i)}\|$ follows as

$$\zeta_i = \frac{\|z_i\|}{\sqrt{2-2\cos\frac{\pi}{n}}}\,.$$

It remains to upper bound the norm of the point to project for all parameter values and all iterations. In fact, from (32)

$$\begin{aligned}\|z_i\| &\leq q^{-1}\|V_{\mathrm{r}}{}^T\|\,\|\lambda_i\| + \|\bar{t}_{\mathrm{r}}\|\,, \quad \text{for all } i \geq 0\\ &\leq 2q^{-1}V_{\mathrm{dc}}\|\overline{\psi_{\mathrm{s,err}}}\|\sqrt{\frac{n}{6}} + \sqrt{3n}\,\bar{t}_{\max} \triangleq \rho(n)\,.\end{aligned} \tag{35}$$

Since modifications of the switching times are cheap to implement, the scalar weight q is in the order of 10^{-4} in our application. As a consequence from (35), the number of integer bits in an implementation depends on the magnitude of the point to be projected as the constant V_{dc} is typically in $[1.5, 2.5]$. In fact, the bound on the largest absolute number follows from the approximate projection on the monotone cone (cf. (25a))

$$\pi_{\mathbb{K}}(z_i) \approx z_i - G^T\eta_{1(i)}\,.$$

Taking into account $\|G^T\|_\infty = 2$ so that

$$\|z_i - G^T\eta_{1(i)}\|_\infty \leq \|z_i\| + 2\zeta\cot^2\!\left(\frac{\pi}{2n}\right),$$

we finally arrive at

$$\#\text{(integer bits)}(n) = \left\lceil \log_2 \rho(n)\left(1 + 2\frac{\cot^2\!\big(\frac{\pi}{2n}\big)}{\sqrt{2-2\cos\frac{\pi}{n}}}\right)\right\rceil$$

for the classic gradient method. Choosing this number of integer bits guarantees that no overflow occurs in fixed-point arithmetic when solving the dual MP^{3}C problem. The actual number of integer bits that result from our analysis for $n \in \{3,4,5\}$ are reported in Table I.

Remark 2. *The number of integer bits for the fast gradient method can be shown to be greater or equal to the number of integer bits for the classic gradient method (cf. Table II). The worst case analysis follows similar reasoning as for the classic gradient method.*

B. Scaling of Iterates for Accuracy and Resource Efficiency

The analysis of the previous section shows that the number of integer bits depends on the magnitude of the point to be projected onto the truncated monotone cone. A closer look at (35) reveals that the largest number inside the gradient methods is about q^{-1} ($\approx 10^4$) times larger than the magnitude of the dual iterate.

In our implementation we aim for a consistent word length[5] and the same number of integer bits and fractional bits for all iterates and intermediate variables of the gradient methods. For this reason, 'up-scaling' the dual iterate is important in order to increase the accuracy given a short fractional bit length. Most importantly, changing the scale of the dual iterate does *not* change the size of the point to be projected. Thus the analysis leading to the number of integer bits in Section IV-A remains valid. By changing the scales of the dual iterates, the number of fractional bits is primarily determined by the accuracy requirement in the primal domain.

When done properly, rescaling of the dual iterates can also reduce the number of DSP-type multiplications on the FPGA.

[5]Word length $\triangleq$ #(integer bits) + #(fractional bits) + 1 (sign bit)

In order to show this for the classic gradient method, let us define the scaled dual iterate $\hat{\lambda}_i$ as

$$\hat{\lambda}_i = 2^b D^{-1} \lambda_i$$

where b is a positive 'upscale integer'. The diagonal matrix D follows from decomposing the matrix V_r in (7) as $V_\text{r} = DU_\text{r}$ with

$$D = \frac{V_\text{dc}}{6} \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{3} \end{bmatrix},$$

$$U_\text{r} = \begin{bmatrix} (2\Delta u_\text{a}, 0_{n-n_\text{a}})^T & (-\Delta u_\text{b}, 0_{n-n_\text{b}})^T & (-\Delta u_\text{c}, 0_{n-n_\text{c}})^T \\ {0_n}^T & (\Delta u_\text{b}, 0_{n-n_\text{b}})^T & (-\Delta u_\text{c}, 0_{n-n_\text{c}})^T \end{bmatrix}.$$

Iteration (32) for the classic gradient method now becomes

$$\hat{\lambda}_{i+1} = \hat{\lambda}_i - \frac{h}{L_d}\Big(\underbrace{\hat{\lambda}_i + 2^b D^{-1}\psi_\text{s,err} - 2^b U_\text{r}\bar{t}_\text{r}}_{\text{expression ①}} \qquad (36)$$
$$+ \underbrace{2^b U_\text{r}\, \pi_{\mathbb{X}_\text{r}}\Big(q^{-1}2^{-b}U_\text{r}^T D^2 \hat{\lambda}_i + \bar{t}_\text{r}\Big)}_{\text{expression ②}} \Big),$$

where $D^2 = (V_\text{dc}/6)^2 \left[\begin{smallmatrix} 1 & 0 \\ 0 & 3 \end{smallmatrix}\right]$. So by scaling the dual iterate, one multiplication with $\sqrt{3}$ disappears. We are left with the following multiplications that require a DSP-type multiplier:

① $D^{-1}\psi_\text{s,err}$

② If $q^{-1}(V_\text{dc}/6)^2 = 2^d$ for an integer d, this expression can be evaluated without any DSP-type multiplier. The assumption is very mild, since q is a tuning parameter and can always be adjusted so that the latter condition holds. Note that multiplication with the factor 3 in D^2 is split up in a multiplication by 2 and an addition.

- One more DSP-type multiplication occurs when the gradient ① + ② is multiplied by the step size h/L_d.

If the gradient is evaluated prior to multiplication by the step size, a rough analysis shows that 'upscale integer' b can be chosen as $b = 5/6/7$ for $n = 3/4/5$ so that no overflow occurs.

C. Estimating the Number of Required Clock Cycles

Before deriving an estimate for the number of clock cycles, we have a look at the total number of DSP-type multipliers needed for an implementation of the gradient methods. As analyzed in the previous section, we have $D^{-1}\psi_\text{s,err}$, multiplication of the (scaled) dual gradient with the step size h/L_d (classic gradient method) or $1/L_d$ (fast gradient method) and the additional multiplication with the step size β_i in the fast gradient method. In all cases, at least one multiplicand has a magnitude of less than one, i.e. it can be represented without any integer bits (in fact, in our application $\|\psi_\text{s,err}\|_\infty < 1$, $h/L_d < 1$, $1/L_d < 1$ and $\beta_i < 1$ for all $i \geq 0$).

In Section I-B we have assumed that our target FPGA provides DSP-type multipliers that are able to multiply 18 bit numbers with each other in one clock cycle (plus two cycles for the input and output registers). As seen from Tables I and II, the word lengths in our implementation are greater than 18 bits in all cases, thus 4 multipliers are required to multiply numbers of word lengths up to 36 bit. However, since all multiplications involve multiplicands of magnitudes below one and the number of necessary fractional bits in our application is always less than 18 bits, only two DSP-type multipliers are required to compute a product in MP^{3}C. Furthermore, we reuse both of these multipliers for all necessary DSP-type multiplications. Hence, a total of only two DSP-type multipliers is required for both gradient methods and all considered problem sizes with a maximum of $n \in \{3, 4, 5\}$ transitions per phase.

The estimates on the number of clock cycles are based upon the following assumptions:

- The multiplication/division by powers of two (left and right shifts respectively), negation, comparison, addition and subtraction have a latency of one clock cycle.
- The multiplication of an 18 bit number with a 36 bit number has a latency of 5 clock cycles. Every additional multiplication has a latency of one cycle (*pipelining*).
- The summation of m numbers takes $\lceil \log_2 m \rceil$ clock cycles (adder tree).

Under these assumptions, it can be shown that the evaluation of the expression

① takes $\max\{6, 1 + \lceil \log_2 3n \rceil\} + 2$ clock cycles,

② takes $11 + \lceil \log_2 3n \rceil$ clock cycles.

Furthermore, the gradient update, i.e. the computation of $\hat{\lambda}_{i+1}$, can be accomplished in 8 cycles. For the fast gradient method, additional 8 cycles are required for the update of the secondary iterate.

Since expression ① can be evaluated in parallel with expression ②, we deduce that 23 cycles are necessary for a single iteration of the classic gradient method and 31 cycles for the fast gradient method. Both estimates hold true for a maximum of $n \in \{3, 4, 5\}$ transitions per phase.

Remark 3. *The estimate for the number of clock cycles depends on the evaluation order of the terms in iteration* (36). *Other evaluation orders than the one presented here are possible and can lead to a further reduction of the number of clock cycles.*

V. Fixed-Point Simulation Results

The proposed gradient methods were implemented in Matlab both in floating-point arithmetic (standard double precision) and fixed-point arithmetic using Fixed Point Designer (vers. 4.2). Note that the reported fixed-point simulation results are identical to the results one would obtain from an actual implementation of our method in hardware.

As an input we used 1700 to 2500 realistic test data samples generated from a three-level neutral point clamped inverter drive system with a medium-voltage induction machine operating at nominal speed and rated torque. This drive system is described in [8] along with its parameters.

A typical device switching frequency of 170 Hz implies that a switching transition in one of the three phases occurs, on average, every 1 ms. An accuracy of 1% of the resulting switching instant is deemed acceptable, which translates into an accuracy of 10 µs. Based on this, we determined the minimum number of iterations i_min for a primal iterate that satisfies $\|\Delta t_{\text{r},i_\text{min}} - \Delta t_\text{r}^*\|_\infty \leq 10$ µs (cf. Section I-B) using floating-point simulations. Herein, $\Delta t_{\text{r},i_\text{min}}$ denotes the approximate

TABLE I
FIXED-POINT SIMULATION RESULTS FOR THE CLASSIC GRADIENT METHOD

Transitions	**Number Format**	**Iterations**	**Accuracy**[1] **[µs]**	**Clock Cycles**
n	integer[2]/fraction[3]	$i_{\min}$	mean/std.dev./max	total
3	14 bits / 13 bits	13	1.59 / 1.13 / 7.87	299
4	16 bits / 14 bits	24	1.00 / 0.84 / 6.54	552
5	17 bits / 14 bits	30	1.08 / 1.21 / 9.14	690

TABLE II
FIXED-POINT SIMULATION RESULTS FOR THE FAST GRADIENT METHOD

Transitions	**Number Format**	**Iterations**	**Accuracy**[1] **[µs]**	**Clock Cycles**
n	integer[2]/fraction[3]	$i_{\min}$	mean/std.dev./max	total
3	15 bits / 13 bits	20	1.25 / 0.96 / 9.00	620
4	16 bits / 15 bits	35	0.27 / 0.45 / 8.50	1085
5	17 bits / 14 bits	35	0.47 / 0.91 / 9.71	1085

[1]The accuracy measure is $\|\Delta t_{\mathrm{r},i_{\min}} - \Delta t_{\mathrm{r}}^*\|_\infty$ where $\Delta t_{\mathrm{r},i_{\min}}$ is the approximate projection on the truncated monotone cone at the final dual iterate.

[2]The number of integer bits follows from the worst case analysis in Section IV-A.

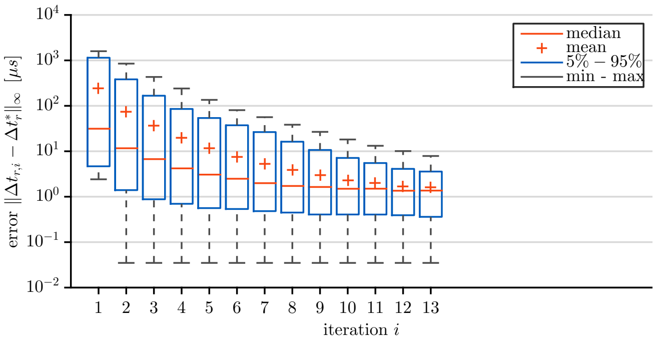
[3]The number of fractional bits was determined empirically as the smallest number such that the required accuracy is reached after $i_{\min}$ iterations in floating point arithmetic.

projection on the truncated monotone cone at the final dual iterate $\hat{\lambda}_{i_{\min}}$ in the iteration given by (36) for the classic gradient method (and similarly for the fast gradient method).

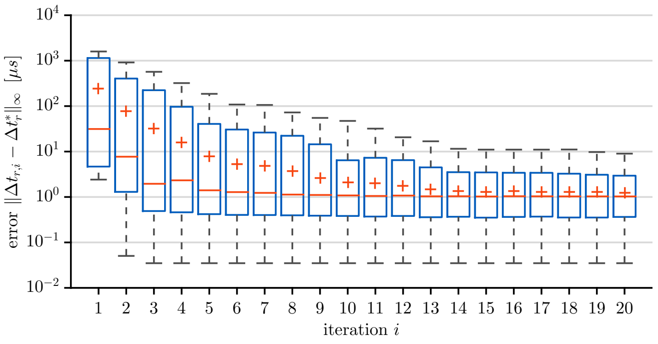
After determining the number of iterations, the minimum number of fractional bits was empirically identified from fixed-point simulations so that the accuracy requirement could be met within the previously determined number of iterations. The number of integer bits was chosen according to the worst case analysis in Section IV-A.

The results are summarized in Table I for the classic gradient method and in Table II for the fast gradient method. The word lengths, including the sign bit, for both gradient methods and all problem sizes do not exceed 32 bits. From the mean accuracy and the standard deviation (abbreviated with std.dev in the tables) it follows that most of the test cases can be solved at a much higher accuracy than required. In order to emphasize this point, Figure 1 illustrates the overall convergence behavior of both gradient methods for the case $n = 3$ as boxplots. We observe that both methods satisfy the accuracy requirement already after ten iterations for more than 95% of the test data samples. The plots also indicate that the classic gradient method (cf. Figure 1a) decreases the worst case error faster than the fast gradient method (cf. Figure 1b), while the fast gradient method reduces the average error quicker. This behavior can be attributed to the nonmonotone behavior of the fast gradient method with respect to the function values along the iterates. Note that in this application, the worst case behavior is more relevant.

Finally, we emphasize that our proposed solution method in the dual domain requires only two DSP-type multipliers, independent of the problem size. Compared to the existing solution method [13], which is based on the fast gradient method in the primal domain and uses about 3000 clock cycles, our approach converges ten times faster and reduces the number



(a) Error convergence of the classic gradient method (full sample set).



(b) Error convergence of the fast gradient method (full sample set).

Fig. 1. Convergence of the largest componentwise error from the optimal solution $\|\Delta t_{r,i} - \Delta t_r^*\|_\infty$ for the full test data sample set in case of $n = 3$ (fixed-point number representation as given in Tables I and II). After ten iterations, both the classic gradient method and the fast gradient method reduce the error below 10 µs for more than 95% of the data samples.

of multipliers by a factor of at least 17 (cf. resource metrics in the footnote on pg. 3). Furthermore, compared to the deadbeat variation of MP^{3}C, see [8], hardware resource usage with respect to the key metric of number of DSP-type multipliers is reduced by a factor three, while the execution time is six times longer. However, in terms of control performance and robustness, our optimization-based approach is much more favorable.

VI. Future Work

Future work could concentrate on reducing the word length. Empirical tests have shown that the number of integer bits can be reduced by up to seven bits without causing overflows. However, excluding overflows for reduced word lengths by means of a rigorous analysis would require more assumptions to be made on the parameters of the MP^{3}C problem.

Another theoretical aspect left for future research is to investigate why the gradient methods converge faster in the dual domain than in the primal domain in this application although the condition numbers are not too different.

References

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

[2] A. K. Rathore, J. Holtz, and T. Boller, "Synchronous optimal pulsewidth modulation for low-switching-frequency control of medium-voltage multilevel inverters," *IEEE Trans. Ind. Electron.*, vol. 57, no. 7, pp. 2374–2381, Jul. 2010.

[3] B. Wu, *High-power converters and AC drives*. New York: Wiley, 2006.

[4] M. P. Kazmierkowski and L. Malesani, “Current control techniques for three-phase voltage-source PWM converters: A survey,” *IEEE Trans. Ind. Electron.*, vol. 45, no. 5, pp. 691–703, Oct. 1998.

[5] B. Beyer, “Schnelle Stromregelung für Hochleistungsantriebe mit Vorgabe der Stromtrajektorie durch off-line optimierte Pulsmuster,” Ph.D. dissertation, Wuppertal University, 1998.

[6] 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.

[7] ——, “Fast dynamic control of medium voltage drives operating at very low switching frequency—An overview,” *IEEE Trans. Ind. Electron.*, vol. 55, no. 3, pp. 1005–1013, Mar. 2008.

[8] T. Geyer, N. Oikonomou, G. Papafotiou, and F. Kieferndorf, “Model Predictive Pulse Pattern Control,” *IEEE Transactions on Industry Applications*, vol. 48, no. 2, pp. 663–676, Apr. 2012.

[9] J. M. Maciejowski, *Predictive control*. Prentice Hall, 2002.

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

[11] E. C. Kerrigan, “Co-design of hardware and algorithms for real-time optimization,” in *European Control Conference (ECC)*, Jun. 2014, pp. 2484–2489.

[12] N. Oikonomou, C. Gutscher, P. Karamanakos, F. D. Kieferndorf, and T. Geyer, “Model predictive pulse pattern control for the five-level active neutral-point-clamped inverter,” *IEEE Transactions on Industry Applications*, vol. 49, no. 6, pp. 2583–2592, 2013.

[13] H. Peyrl, J. Liu, and T. Geyer, “An FPGA implementation of the fast gradient method for solving the Model Predictive Pulse Pattern Control problem,” in *IEEE International Symposium on Sensorless Control for Electrical Drives and Predictive Control of Electrical Drives and Power Electronics (SLED/PRECEDE)*, 2013, pp. 1–6.

[14] Y. Nesterov, *Introductory Lectures on Convex Optimization. A Basic Course*. Springer, 2004.

[15] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, “The Explicit Linear Quadratic Regulator for Constrained Systems,” *Automatica*, vol. 38, no. 1, pp. 3–20, Jan. 2002.

[16] A. Suardi, S. Longo, E. Kerrigan, and G. Constantinides, “Energy-aware MPC co-design for DC-DC converters,” in *European Control Conference (ECC)*, Jul. 2013, pp. 3608–3613.

[17] M. Herceg, M. Kvasnica, C. Jones, and M. Morari, “Multi-Parametric Toolbox 3.0,” in *Proceedings of the European Control Conference*, Zürich, Switzerland, July 17–19 2013, pp. 502–510, http://control.ee.ethz.ch/~mpt.

[18] P. Patrinos, A. Guiggiani, and A. Bemporad, “A dual gradient-projection algorithm for model predictive control in fixed-point arithmetic,” *Automatica*, vol. 55, pp. 226–235, May 2015.

[19] M. Rubagotti, P. Patrinos, A. Guiggiani, and A. Bemporad, “Real-time model predictive control based on dual gradient projection: Theory and fixed-point FPGA implementation,” *International Journal of Robust and Nonlinear Control*, p. n/a, Jan. 2016.

[20] H. Peyrl, A. Zanarini, T. Besselmann, J. Liu, and M.-A. Boéchat, “Parallel implementations of the fast gradient method for high-speed MPC,” *Control Engineering Practice*, vol. 33, pp. 22–34, Dec. 2014.

[21] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, Massachusetts: Athena Scientific, 1999.

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

[23] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York: Springer, 2006.

[24] J. Mattingley and S. Boyd, “CVXGEN: A code generator for embedded convex optimization,” *Optimization and Engineering*, vol. 13, no. 1, pp. 1–27, 2012.

[25] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, “qpOASES: A parametric active-set algorithm for quadratic programming,” *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[26] O. Devolder, F. Glineur, and Y. Nesterov, “First-order methods of smooth convex optimization with inexact oracle,” *Mathematical Programming*, vol. 146, no. 1-2, pp. 37–75, 2014.

[27] ——, “First-order methods with inexact oracle: the strongly convex case,” Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), Tech. Rep., 2013.

[28] S. Richter, C. N. Jones, and M. Morari, “Certification aspects of the fast gradient method for solving the dual of parametric convex programs,” *Mathematical Methods of Operations Research*, vol. 77, no. 3, pp. 305–321, Dec. 2012.

[29] D. P. Bertsekas, *Convex Optimization Theory*, 1$^{\text{st}}$ ed. Athena Scientific, Jun. 2009.

[30] A. B. Németh and S. Z. Németh, “How to project onto the monotone nonnegative cone using Pool Adjacent Violators type algorithms,” *arXiv:1201.2343 [math, stat]*, Jan. 2012, arXiv: 1201.2343.

[31] A. Németh and S. Németh, “How to project onto an isotone projection cone,” *Linear Algebra and its Applications*, vol. 433, no. 1, pp. 41–51, Jul. 2010.

[32] S. Richter, “Computational complexity certification of gradient methods for real-time model predictive control,” Ph.D. dissertation, ETH Zürich, Zürich, 2012.

[33] P. Giselsson, “Improved fast dual gradient methods for embedded model predictive control,” in *Proceedings of 2014 IFAC World Congress*, 2014.

[34] L. E. Ghaoui, E. Feron, V. Balakrishnan, and S. Boyd, *Linear Matrix Inequalities in System & Control Theory*. Society for Industrial and Applied Mathematics, Jul. 1994.

[35] A. B. Taylor, J. M. Hendrickx, and F. Glineur, “Smooth Strongly Convex Interpolation and Exact Worst-case Performance of First-order Methods,” *arXiv:1502.05666 [math]*, Feb. 2015.