High-Speed Finite Control Set Model Predictive Control for Power Electronics

Bartolomeo Stellato, *Student Member, IEEE,* Tobias Geyer, *Senior Member, IEEE* and Paul J. Goulart, *Member, IEEE*

***Abstract*—Common approaches for direct model predictive control (MPC) for current reference tracking in power electronics suffer from the high computational complexity encountered when solving integer optimal control problems over long prediction horizons. We propose an efficient alternative method based on approximate dynamic programming, greatly reducing the computational burden and enabling sampling times below $25\ \mu$s. Our approach is based on the offline estimation of an infinite horizon value function which is then utilized as the tail cost of an MPC problem. This allows us to reduce the controller horizon to a very small number of stages while simultaneously improving the overall controller performance. Our proposed algorithm was implemented on a small size FPGA and validated on a variable speed drive system with a three-level voltage source converter. Time measurements showed that our algorithm requires only $5.76\ \mu$s for horizon $N = 1$ and $17.27\ \mu$s for $N = 2$, in both cases outperforming state of the art approaches with much longer horizons in terms of currents distortion and switching frequency. To the authors' knowledge, this is the first time direct MPC for current control has been implemented on an FPGA solving the integer optimization problem in real-time and achieving comparable performance to formulations with long prediction horizons.**

***Index Terms*—Approximate dynamic programming, value function approximation, drive systems, finite control set, model predictive control (MPC).**

I. Introduction

AMONG the control strategies adopted in power electronics, model predictive control (MPC) [1] has recently gained popularity due to its various advantages [2]. MPC has been shown to outperform traditional control methods mainly because of its ease in handling time-domain constraint specifications that can be imposed by formulating the control problem as a constrained optimization problem. Due to its structure, MPC can be applied to a variety of power electronics topologies and operating conditions providing a higher degree of flexibility than traditional approaches.

With the recent advances in convex optimization techniques [3], it has been possible to apply MPC to very fast constrained linear systems with continuous inputs by solving convex quadratic optimization problems within microseconds [4]. However, when dealing with nonlinear systems or with integer inputs, the optimal control problems are no longer convex and it is harder to find optimal solutions. Sequential quadratic programming (SQP) [3] has gained popularity because of its ease in iteratively approximating nonconvex continuous control problems as convex quadratic programs, which can be solved efficiently. Moreover, integrated perturbation analysis (IPA) has been recently combined with SQP methods (IPA-SQP) [5] and applied to power electronics [6] by solving the approximated quadratic optimization problem at a given time instant using a perturbed version of the problem at the previous time instant, thereby reducing the number of iterations required at each time step. However, there are still two orders of magnitude difference in achievable computation time compared to results obtained for linear systems [4] and further advances are required to apply these methods to very fast dynamical systems.

In power electronics, many conventional control strategies applied in industry are based on proportional-plus-integral (PI) controllers providing continuous input signals to a modulator that manages conversion to discrete switch positions. Direct MPC [7] instead combines the current control and the modulation problem into a single computational problem, providing a powerful alternative to conventional PI controllers. With direct MPC, the manipulated variables are the switch positions, which lie in a discrete and finite set, giving rise to a switched system. Therefore, this approach does not require a modulator and is often referred to as *finite control set* MPC.

Since the manipulated variables are restricted to be integer-valued, the optimization problem underlying direct MPC is provably $\mathcal{NP}$-hard [8]. In power electronics these optimization problems are often solved by complete enumeration of all the possible solutions, which grow exponentially with the prediction horizon [9]. Since long horizons are required to ensure stability and good closed-loop performance [10], direct MPC quickly becomes intractable for real-time applications. As a consequence, in cases when reference tracking of the converter currents is considered, the controller horizon is often restricted to one [2]. Recently, the IPA-SQP method has been applied to a finite control set MPC [11] to efficiently approximate the optimization problem in the case of nonlinear systems. However, no particular attention is paid to reducing the number of integer combinations that must be evaluated, which is at the source of the most significant computational issues. Despite attempts to overcome the computational burden of integer programs in power electronics [12], the problem of implementing these algorithms on embedded systems remains open.

This work was supported by the People Programme (Marie Curie Actions) of the European Unions Seventh Framework Programme (FP7/2007-2013) under REA grant agreement no 607957 (TEMPO).

B. Stellato and P.J. Goulart are with the University of Oxford, Parks Road, Oxford, OX1 3PJ, U.K. (e-mail: bartolomeo.stellato@eng.ox.ac.uk; paul.goulart@eng.ox.ac.uk).

T. Geyer is with the ABB Corporate Research Centre, 5405 Baden-Dattwil, Switzerland (e-mail: t.geyer@ieee.org).

A recent technique introduced in [13] and benchmarked in [14] reduces the computational burden of direct MPC when increasing the prediction horizon. In that work the optimization problem was formulated as an integer least-squares (ILS) problem and solved using a tailored branch-and-bound algorithm, described as *sphere decoding* [15], generating the optimal switching sequence at each time step. Although this approach appears promising relative to previous work, the computation time required to perform the sphere decoding algorithm for long horizons (i.e. $N = 10$), is still far slower than the sampling time typically required, i.e. $T_s = 25\,\mu$s. In the very recent literature, some approaches have been studied to improve the computational efficiency of the sphere decoding algorithm. In particular, in [16] a method based on a lattice reduction algorithm decreased the average computational burden of the sphere decoding. However, the worst case complexity of this new reformulation is still exponential in the horizon length. In [17], heuristic search strategies for the sphere decoding algorithm are studied at the expense of suboptimal control performance. Even though a floating point complexity analysis of the algorithms is presented in these works, no execution times and no details about fixed-point implementation are provided. Furthermore, there currently exists no embedded implementation of a direct MPC algorithm for current control achieving comparable performance to formulations with long prediction horizons.

This paper introduces a different method to deal with the direct MPC problem. In contrast to common formulations [18] where the switching frequency is controlled indirectly via penalization of the input switches over the controller horizon, in this work the system dynamics are augmented to directly estimate the switching frequency. Our approach allows the designer to set the desired switching frequency a priori by penalizing its deviations from this estimate. Thus, the cost function tuning can be performed more easily than with the approach in [14] and [13], where a tuning parameter spans the whole frequency range with no intuitive connection to the desired frequency. To address the computational issues of long prediction horizons, we formulate the tracking problem as a regulation one by augmenting the state dynamics and cast it in the framework of approximate dynamic programming (ADP) [19]. The infinite horizon value function is approximated using the approach in [20] and [21] by solving a semidefinite program (SDP) [22] offline. This enables us to shorten the controller horizon by applying the estimated tail cost to the last stage to maintain good control performance. In [23] the authors applied a similar approach to stochastic systems with continuous inputs, denoting the control law as the "iterated greedy policy".

As a case study, our proposed approach is applied to a variable-speed drive system consisting of a three-level neutral point clamped voltage source inverter connected to a medium-voltage induction machine. The plant is modelled as a linear system with a switched three-phase input with equal switching steps for all phases.

Closed loop simulations in MATLAB in steady state operation showed that with our method even very short prediction horizons give better performance than the approach in [14] and [13] with much longer planning horizons.

We have implemented our algorithm on a small size Xilinx Zynq FPGA (xc7z020) in fixed-point arithmetic and verified its performance with hardware in the loop (HIL) tests of both steady-state and transient performance. The results achieve almost identical performance to closed-loop simulations and very fast computation times, allowing us to comfortably run our controller within the $25\,\mu$s sampling time.

The remainder of the paper is organized as follows. In Section II we describe the drive system case study and derive the physical model. In Section III the direct MPC problem is derived by augmenting the state dynamics and approximating the infinite horizon tail cost using ADP. Section IV describes all of the physical parameters of the model used to verify our approach. In Section V we present closed-loop simulation results on the derived model in steady state operation to characterize the achievable performance of our method. In Section VI we describe the hardware setup, the algorithm and all the FPGA implementation details. In Section VII HIL tests are performed in steady-state and transient operation. Finally, we provide conclusions in Section VIII.

In this work we use normalized quantities by adopting the per unit system (pu). The time scale t is also normalized using the base angular velocity ω_b that in this case is $2\pi 50\,\mathrm{rad/s}$, i.e. one time unit in the per unit system corresponds to $1/\omega_b\,\mathrm{s}$. Variables in the three-phase system (*abc*) are denoted $\boldsymbol{\xi}_{abc} = \begin{bmatrix}\xi_a & \xi_b & \xi_c\end{bmatrix}^\top$. It is common practice to transform phase variables to $\boldsymbol{\xi}_{\alpha\beta}$ in the stationary orthogonal $\alpha\beta$ coordinates by $\boldsymbol{\xi}_{\alpha\beta} = \boldsymbol{P}\boldsymbol{\xi}_{abc}$. The inverse operation can be performed as $\boldsymbol{\xi}_{abc} = \boldsymbol{P}^\dagger\boldsymbol{\xi}_{\alpha\beta}$. The matrices $\boldsymbol{P}$ and $\boldsymbol{P}^\dagger$ are the Clarke transform and its pseudoinverse respectively, i.e.

$$\boldsymbol{P} = \frac{2}{3}\begin{bmatrix}1 & -\frac{1}{2} & -\frac{1}{2}\\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2}\end{bmatrix}, \quad \boldsymbol{P}^\dagger = \begin{bmatrix}1 & 0\\ -\frac{1}{2} & \frac{\sqrt{3}}{2}\\ -\frac{1}{2} & -\frac{\sqrt{3}}{2}\end{bmatrix}.$$

II. Drive System Case Study

In this work we consider a variable-speed drive system as shown in Figure 1, consisting of a three-level neutral point clamped (NPC) voltage source inverter driving a medium-voltage (MV) induction machine. The total dc-link voltage V_{dc} is assumed constant and the neutral point potential N fixed.

In most modern approaches to control variable-speed drive systems, the control is split into two cascaded loops. The outer loop controls the machine speed by manipulating the torque reference. The inner loop controls the torque and the fluxes by manipulating the voltages applied to the stator windings of the machine. Our approach focuses on the inner loop. The reference torque is converted into stator currents references that must be tracked and the controller manipulates the stator voltages by applying different inverter switch positions.

A. Physical Model of the Inverter

The switch positions in the three phase legs can be described by the integer input variables $u_a, u_b, u_c \in \{-1, 0, 1\}$, leading

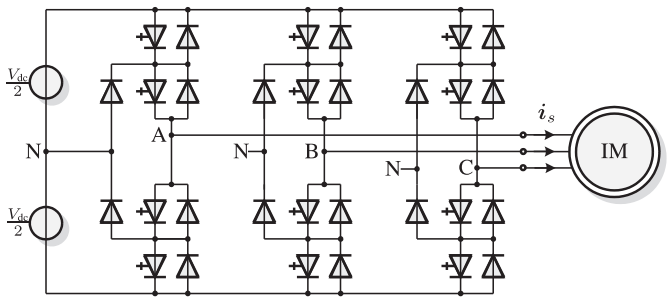


Fig. 1. Three-level three-phase neutral point clamped (NPC) voltage source inverter driving an induction motor with a fixed neutral point potential.

to phase voltages $\{-\frac{V_{dc}}{2}, 0, \frac{V_{dc}}{2}\}$, respectively. Hence, the output voltage of the inverter is given by

$$\boldsymbol{v}_{\alpha\beta} = \frac{V_{dc}}{2}\boldsymbol{u}_{\alpha\beta} = \frac{V_{dc}}{2}\boldsymbol{P}\boldsymbol{u}_{sw}, \tag{1}$$

where $\boldsymbol{u}_{sw} = \begin{bmatrix} u_a & u_b & u_c \end{bmatrix}^\top$.

B. Physical Model of the Machine

Hereafter we derive the state-space model of the squirrel-cage induction machine in the $\alpha\beta$ plane. Since we are considering a current control problem, it is convenient to use the stator current $\boldsymbol{i}_{s,\alpha\beta}$ and the rotor flux $\boldsymbol{\psi}_{r,\alpha\beta}$ as state variables. The model input is the stator voltage $\boldsymbol{v}_{s,\alpha\beta}$ which is equal to the inverter output voltage in (1). The model parameters are: the stator and rotor resistances R_s and R_r; the mutual, stator and rotor reactances X_m, X_{ls} and X_{lr}, respectively; the inertia J; and the mechanical load torque T_l. Given these quantities, the continuous-time state equations [24], [25] are

$$\begin{aligned}
\frac{\mathrm{d}\boldsymbol{i}_s}{\mathrm{d}t} &= -\frac{1}{\tau_s}\boldsymbol{i}_s + \left(\frac{1}{\tau_r}\boldsymbol{I} - \omega_r \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\right)\frac{X_m}{D}\boldsymbol{\psi}_r + \frac{X_r}{D}\boldsymbol{v}_s \\
\frac{\mathrm{d}\boldsymbol{\psi}_r}{\mathrm{d}t} &= \frac{X_m}{\tau_r}\boldsymbol{i}_s - \frac{1}{\tau_r}\boldsymbol{\psi}_r + \omega_r \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\boldsymbol{\psi}_r \\
\frac{\mathrm{d}\omega_r}{\mathrm{d}t} &= \frac{1}{J}\left(T_e - T_l\right),
\end{aligned} \tag{2}$$

where $D \coloneqq X_s X_r - X_m^2$ with $X_s \coloneqq X_{ls} + X_m$ and $X_r \coloneqq X_{lr} + X_m$, and $\boldsymbol{I}$ represents the 2×2 identity matrix. To simplify the notation, we have dropped the subscripts $\alpha\beta$ from all vectors in (2). Moreover, $\tau_s \coloneqq X_r D / \left(R_s X_r^2 + R_r X_m^2\right)$ and $\tau_r \coloneqq X_r / R_r$ are the transient stator and the rotor time constants respectively. The electromagnetic torque is given by

$$T \coloneqq \frac{X_m}{X_r}\left(\boldsymbol{\psi}_r \times \boldsymbol{i}_s\right). \tag{3}$$

The rotor speed ω_r is assumed to be constant within the prediction horizon. For prediction horizons in the order of a few milliseconds this is a mild assumption.

C. Complete Model of the Physical System

Given the models of the drive and of the induction motor in (1) and (2) respectively, the state-space model in the continuous time domain can be described as

$$\frac{\mathrm{d}\boldsymbol{x}_{ph}(t)}{\mathrm{d}t} = \boldsymbol{D}\boldsymbol{x}_{ph}(t) + \boldsymbol{E}\boldsymbol{u}_{sw}(t) \tag{4a}$$

$$\boldsymbol{y}_{ph}(t) = \boldsymbol{F}\boldsymbol{x}_{ph}(t), \tag{4b}$$

where the state vector $\boldsymbol{x}_{ph} = \begin{bmatrix} i_{s,\alpha} & i_{s,\beta} & \psi_{r,\alpha} & \psi_{r,\beta} \end{bmatrix}^\top$ includes the stator current and rotor flux in the $\alpha\beta$ reference frame. The output vector is taken as the stator current, i.e. $\boldsymbol{y}_{ph} = \boldsymbol{i}_{s,\alpha\beta}$. The matrices $\boldsymbol{D}, \boldsymbol{E}$ and $\boldsymbol{F}$ are defined in Appendix A.

The state-space model of the drive can be converted into the discrete-time domain using exact Euler discretization. By integrating (4a) from $t = k\hat{T}_s$ to $t = (k+1)\hat{T}_s$ and keeping $\boldsymbol{u}_{sw}(t)$ constant during each interval and equal to $\boldsymbol{u}_{sw}(k)$, the discrete-time model becomes

$$\boldsymbol{x}_{ph}(k+1) = \boldsymbol{A}_{ph}\boldsymbol{x}_{ph}(k) + \boldsymbol{B}_{ph}\boldsymbol{u}_{sw}(k) \tag{5a}$$

$$\boldsymbol{y}_{ph}(k) = \boldsymbol{C}_{ph}\boldsymbol{x}_{ph}(k), \tag{5b}$$

with matrices $\boldsymbol{A}_{ph} \coloneqq e^{\boldsymbol{D}\hat{T}_s}$, $\boldsymbol{B}_{ph} \coloneqq -\boldsymbol{D}^{-1}\left(\boldsymbol{I} - \boldsymbol{A}_{ph}\right)\boldsymbol{E}$, $\boldsymbol{C}_{ph} \coloneqq \boldsymbol{F}$ and $k \in \mathbb{N}$. $\boldsymbol{I}$ is an identity matrix of appropriate dimensions. Although the sampling time is $T_s = 25\,\mu$s, we use the discretization interval $\hat{T}_s = T_s\omega_b$ for consistency with our per unit system, where ω_b is the base frequency.

III. Model Predictive Current Control

A. Problem Description

Our control scheme must address two conflicting objectives simultaneously. On the one hand, the distortion of the stator currents $\boldsymbol{i}_s$ cause iron and copper losses in the machine leading to thermal losses. Because of the limited cooling capability of the electrical machine, the stator current distortions have to be kept as low as possible. On the other hand, high frequency switching of the inputs $\boldsymbol{u}_{sw}$ produces high power losses and stress on the semiconductor devices. Owing to the limited cooling capability in the inverter, we therefore should minimize the switching frequency of the integer inputs.

Note that the effect of the inverter switchings on the torque ripples can be improved during the machine design. In particular, increasing the time constants of the stator and the rotor τ_s and τ_r can reduce the amplitude of the torque ripples by decreasing the derivative of the currents $\boldsymbol{i}_s$ and fluxes $\boldsymbol{\varphi}_r$. This is achieved naturally when dealing with machines with higher power. Thanks to the flexibility of model based controller designs such as MPC, different machine dynamics influencing the torque ripples are automatically taken into account by the controller, which adapts the optimal inputs computation depending on the plant parameters. Thus, any improvements during the machine design can be optimally exploited by adapting the internal model dynamics in the controller. Another similar approach is to include LCL filters between the inverter and the motor to decrease the high frequency components of the currents; see [26]. These approaches allow operation at lower switching frequencies with low THD at the same time. However, it is sometimes impossible to change the machine's physical configuration and it is necessary to operate the inverter at high switching frequencies to satisfy high performance requirements in terms of stator currents

distortion. For all these reasons there is an unavoidable tradeoff between these two criteria.

The controller sampling time plays an important role in the distortion and switching frequency tradeoff. Depending on the precision required in defining the inverter switching times, the controller is discretized with higher (e.g. $125\,\mu$s) or lower (e.g. $25\,\mu$s) sampling times. Higher sampling times define a more coarse discretization grid leading to less precise definition of the switching instants, but more available time to perform the computations during the closed-loop cycles. Lower sampling times, on the other hand, lead to improved controller accuracy while reducing the allowed computing time. However, for the same switching frequency, longer sampling times produce higher distortions. Ideally, the sampling time should be chosen as low as possible to have the highest possible accuracy.

In contrast to the common approaches in direct MPC where the switching frequency is minimized, in this work we penalize its difference from the desired frequency which is denoted by f_{sw}^*. This is motivated by the fact that inverters are usually designed to operate at a specific nominal switching frequency.

The current distortion is measured via the total harmonic distortion (THD). Given an infinitely long time-domain current signal i and its fundamental component i^* of constant magnitude, the THD is proportional to the root mean square (RMS) value of the difference $i - i^*$. Hence, we can write for one phase current

$$THD \sim \lim_{M\to\infty} \sqrt{\frac{1}{M}\sum_{k=0}^{M-1}(i(k)-i^*(k))^2}, \tag{6}$$

with $M \in \mathbb{N}$. For the three-phase current $\boldsymbol{i}_{abc}$ and its reference $\boldsymbol{i}_{abc}^*$ the THD is proportional to the mean value of (6) over the phases. It is of course not possible to calculate the THD in real time within our controller computations because of finite storage constraints.

The switching frequency of the inverter can be identified by computing the average frequency of each active semiconductor device. As displayed in Figure 1, the total number of switches in all three phases is 12, and for each switching transition by one step up or down in a phase one semiconductor device is turned on. Thus, the number of *on* transitions occurring between time step $k-1$ and k is given by the 1-norm of the difference of the inputs vectors: $\|\boldsymbol{u}_{sw}(k) - \boldsymbol{u}_{sw}(k-1)\|_1$.

Given a time interval centered at the current time step k from $k-M$ to $k+M$, it is possible to estimate the switching frequency by counting the number of *on* transitions over the time interval and dividing the sum by the interval's length $2MT_s$. We then can average over all the semiconductor switches by dividing the computed fraction by 12. At time k, the switching frequency estimate can be written as

$$f_{sw,M}(k) \coloneqq \frac{1}{12\cdot 2MT_s}\sum_{i=-M}^{M}\|\boldsymbol{u}_{sw}(k+i)-\boldsymbol{u}_{sw}(k+i-1)\|_1, \tag{7}$$

which corresponds to a non-causal finite impulse response (FIR) filter of order $2M$. The true average switching frequency is the limit of this quantity as the window length goes to infinity

$$f_{sw} \coloneqq \lim_{M\to\infty} f_{sw,M}(k), \tag{8}$$

and does not depend on time k.

The f_{sw} computation brings similar issues as the THD. In addition to finite storage constraints, the part of the sum regarding the future signals produces a non-causal filter that is impossible to implement in a real-time control scheme.

These issues in computing THD and f_{sw} are addressed in the following two sections via augmentation of our state space model to include suitable approximation schemes for both quantities.

B. Total Harmonic Distortion

According to (6), the THD in the three-phase current is proportional to the mean value of $(i_{s,a}-i_{s,a}^*)^2+(i_{s,b}-i_{s,b}^*)^2+(i_{s,c}-i_{s,c}^*)^2$. As shown in [13], the THD is also proportional to the stator current ripple in the $\alpha\beta$ coordinate system, i.e.

$$THD \sim \lim_{M\to\infty}\sum_{k=0}^{M-1}\|\boldsymbol{e_i}(k)\|_2^2, \tag{9}$$

where we have introduced the error signal $\boldsymbol{e_i}(k) \coloneqq \boldsymbol{i}_{s,\alpha\beta}(k) - \boldsymbol{i}_{s,\alpha\beta}^*(k)$. It is straightforward to show [27] that the stator current reference during steady-state operation at rated frequency is given by

$$\boldsymbol{i}_{s,\alpha\beta}^*(k) = \begin{bmatrix}\sin(k) & -\cos(k)\end{bmatrix}^\top. \tag{10}$$

Hence, in order to minimize the THD, we minimize the squared 2-norm of vector $\boldsymbol{e_i}$ over all future time steps. We also introduce a discount factor $\gamma \in (0,1)$ to normalize the summation preventing it from going to infinity due to persistent tracking errors. The cost function related to THD minimization is therefore

$$\sum_{k=0}^{\infty}\gamma^k\|\boldsymbol{e_i}(k)\|_2^2. \tag{11}$$

In order to construct a regulation problem, we include the oscillating currents from (10) as two additional uncontrollable states $\boldsymbol{x}_{osc} = \boldsymbol{i}_{s,\alpha\beta}^*$ within our model of the system dynamics. The ripple signal $\boldsymbol{e_i}(k)$ is then modeled as an output defined by the difference between two pairs of system states.

C. Switching Frequency

To overcome the difficulty of dealing with the filter in (7), we consider only the past input sequence, with negative time shift giving a causal FIR filter estimating f_{sw}. This filter is approximated with an infinite impulse response (IIR) one whose dynamics can be modeled as a linear time invariant (LTI) system. Note that future input sequences in (7) are taken into account inside the controller prediction.

Let us define three binary phase inputs denoting whether each phase switching position changed at time k or not, i.e.

$$\boldsymbol{p}(k) \coloneqq \begin{bmatrix}p_a(k) & p_b(k) & p_c(k)\end{bmatrix}^\top \in \{0,1\}^3, \tag{12}$$

with

$$p_s(k) = \|u_s(k) - u_s(k-1)\|_1,\ s \in \{a,b,c\}. \tag{13}$$

It is straightforward to show that the following second order IIR filter will approximate the one-sided version of the FIR filter in (7) [28]:

$$\boldsymbol{x}_{flt}(k+1) = \underbrace{\begin{bmatrix} a_1 & 0 \\ 1-a_1 & a_2 \end{bmatrix}}_{\boldsymbol{A_{flt}}} \boldsymbol{x}_{flt}(k) + \underbrace{\frac{1-a_2}{12T_s}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}}_{\boldsymbol{B}_{flt}} \boldsymbol{p}(k) \tag{14}$$

$$\hat{f}_{sw}(k) = \begin{bmatrix} 0 & 1 \end{bmatrix} \boldsymbol{x}_{flt}(k), \tag{15}$$

where $\hat{f}_{sw}(k)$ is the estimated switching frequency and $\boldsymbol{x}_{flt}(k)$ is the filter state. The two poles at $a_1 = 1 - 1/r_1$ and $a_2 = 1 - 1/r_2$ with $r_1, r_2 >> 0$ can be tuned to shape the behavior of the filter. Increasing a_1, a_2 make the estimate smoother, while decreasing a_1, a_2 gives a faster estimation with more noisy values.

We denote the difference between the approximation $\hat{f}_{sw}(k)$ and the target frequency f^*_{sw} by $e_{sw}(k) := \hat{f}_{sw}(k) - f^*(k)$. Therefore, the quantity to be minimized in order to bring the switching frequency estimate as close to the target as possible is

$$\sum_{k=0}^{\infty} \delta \gamma^k \left\| e_{sw}(k) \right\|_2^2, \tag{16}$$

where $\delta \in \mathbb{R}_+$ is a design parameter included to reflect the importance of this part of the cost relative to the THD component.

Finally, we can augment the state space to include the filter dynamics and the target frequency by adding the states $\begin{bmatrix} \boldsymbol{x}_{flt}^\top & f^*_{sw} \end{bmatrix}^\top$ so that the control inputs try to drive the difference between two states to zero. Since the physical states are expressed in the per unit (pu) system with values around 1, in order to have these augmented states within the same order of magnitude we will normalize them by the desired frequency f^*_{sw} defining $\boldsymbol{x}_{sw} = \begin{bmatrix} (1/f^*_{sw})\boldsymbol{x}_{flt}^\top & 1 \end{bmatrix}$ and the matrices $\boldsymbol{A}_{sw} = \text{blkdiag}(\boldsymbol{A}_{sw}, 1)$, $\boldsymbol{B}_{sw} = \begin{bmatrix} \boldsymbol{B}_{flt}^\top & \boldsymbol{0}_{1\times 3}^\top \end{bmatrix}^\top$.

D. MPC Problem Formulation

Let us define the complete augmented state as

$$\boldsymbol{x}(k) := \begin{bmatrix} \boldsymbol{x}_{ph}(k)^\top & \boldsymbol{x}_{osc}(k)^\top & \boldsymbol{x}_{sw}(k)^\top & \boldsymbol{u}_{sw}(k-1)^\top \end{bmatrix}^\top, \tag{17}$$

with $\boldsymbol{x}(k) \in \mathbb{R}^9 \times \{-1, 0, 1\}^3$ and total state dimension $n_x = 12$. The vector $\boldsymbol{x}_{ph}$ represents the physical system from Section II-C, $\boldsymbol{x}_{osc}$ defines the oscillating states of the sinusoids to track introduced in Section III-B, $\boldsymbol{u}_{sw}(k-1)$ are additional states used to keep track of the physical switch positions at the previous time step, and $\boldsymbol{x}_{sw}$ are the states related to the switching filter from Section III-C.

The system inputs are defined as

$$\boldsymbol{u}(k) := \begin{bmatrix} \boldsymbol{u}_{sw}(k)^\top & \boldsymbol{p}(k)^\top \end{bmatrix}^\top \in \mathbb{R}^{n_u},$$

where $\boldsymbol{u}_{sw}$ are the physical switch positions and $\boldsymbol{p}$ are the three binary inputs entering in the frequency filter from Section III-C. The input dimension is $n_u = 6$. To simplify the notation let us define the matrices $\boldsymbol{G}$ and $\boldsymbol{T}$ to obtain $\boldsymbol{u}_{sw}(k)$ and $\boldsymbol{p}(k)$ from $\boldsymbol{u}(k)$ respectively, i.e. such that $\boldsymbol{u}_{sw}(k) = \boldsymbol{G}\boldsymbol{u}(k)$ and $\boldsymbol{p}(k) = \boldsymbol{T}\boldsymbol{u}(k)$. Similarly, to obtain $\boldsymbol{u}_{sw}(k-1)$ from $\boldsymbol{x}(k)$ we define a matrix $\boldsymbol{W}$ so that $\boldsymbol{u}_{sw}(k-1) = \boldsymbol{W}\boldsymbol{x}(k)$.

The MPC problem with horizon $N \in \mathbb{N}$ can be written as

$$\begin{aligned} \underset{\boldsymbol{u}(k)}{\text{minimize}} \quad & \sum_{k=0}^{N-1} \gamma^k \ell(\boldsymbol{x}(k)) + \gamma^N V(\boldsymbol{x}(N)) && \text{(18a)} \\ \text{subject to} \quad & \boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) && \text{(18b)} \\ & \boldsymbol{x}(0) = \boldsymbol{x}_0 && \text{(18c)} \\ & \boldsymbol{x}(k) \in \mathcal{X},\ \boldsymbol{u}(k) \in \mathcal{U}(\boldsymbol{x}(k)), && \text{(18d)} \end{aligned}$$

where the stage cost is defined combining the THD and the switching frequency penalties in (11) and (16) respectively as

$$\ell(\boldsymbol{x}(k)) = \left\| \boldsymbol{C}\boldsymbol{x}(k) \right\|_2^2 = \left\| \boldsymbol{e_i}(k) \right\|_2^2 + \delta \left\| e_{sw}(k) \right\|_2^2.$$

The tail cost $V(\boldsymbol{x}(N))$ is an approximation of the infinite horizon tail that we will compute in the next section using approximate dynamic programming (ADP). The matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ define the extended system dynamics and the output vector; they can be derived directly from the physical model (5) and from the considerations in Sections III-C and III-B.

The input constraint set is defined as

$$\begin{aligned} \mathcal{U}(\boldsymbol{x}(k)) := & \{ \left\| \boldsymbol{T}\boldsymbol{u}(k) \right\|_\infty \leq 1, && \text{(19a)} \\ & -\boldsymbol{T}\boldsymbol{u}(k) \leq \boldsymbol{G}\boldsymbol{u}(k) - \boldsymbol{W}\boldsymbol{x}(k) \leq \boldsymbol{T}\boldsymbol{u}(k), && \text{(19b)} \\ & \boldsymbol{G}\boldsymbol{u}(k) \in \{-1, 0, 1\}^3 \}, && \text{(19c)} \end{aligned}$$

where constraint (19b) defines the relationship between $\boldsymbol{u}_{sw}$ and $\boldsymbol{p}$ from (12) and (13). Constraint (19a) together with (19b) defines the switching constraints $\left\| \boldsymbol{u}_{sw}(k) - \boldsymbol{u}_{sw}(k-1) \right\|_\infty \leq 1$ imposed to avoid a shoot-through in the inverter positions that could damage the components. Finally, (19c) enforces integrality of the switching positions.

It is straightforward to confirm that the number of switching sequence combinations grows exponentially with the horizon length N, i.e. $3^{3N} = 27^N$. The problem therefore becomes extremely difficult to solve for even modest horizon lengths.

Observe that the controller tuning parameters are δ, which defines the relative importance of the THD and f_{sw} components of the cost function, and r_1, r_2, which shape the switching frequency estimator.

E. Control Loop

The complete block diagram is shown in Figure 2. The desired torque T^* determines the currents $\boldsymbol{x}_{osc}$ by setting the initial states of the oscillator OSC. The motor speed ω_r and the stator currents $\boldsymbol{i}_s$ are measured directly from the machine and used by the observer OBS providing the physical states of the motor $\boldsymbol{x}_{ph}$. The auxiliary inputs $\boldsymbol{p}$ are fed into the filter FLT estimating the switching frequency in $\boldsymbol{x}_{sw}$. The switch positions $\boldsymbol{u}_{sw}$ go through a one step delay and are exploited again by the MPC formulation.

Following a receding horizon control strategy, at each stage k the problem (18) is solved, obtaining the optimal sequence $\{\boldsymbol{u}^\star(k)\}_{k=0}^{N-1}$ from which only $\boldsymbol{u}^\star(0)$ is applied to the switches. At the next stage $k+1$, given new vectors

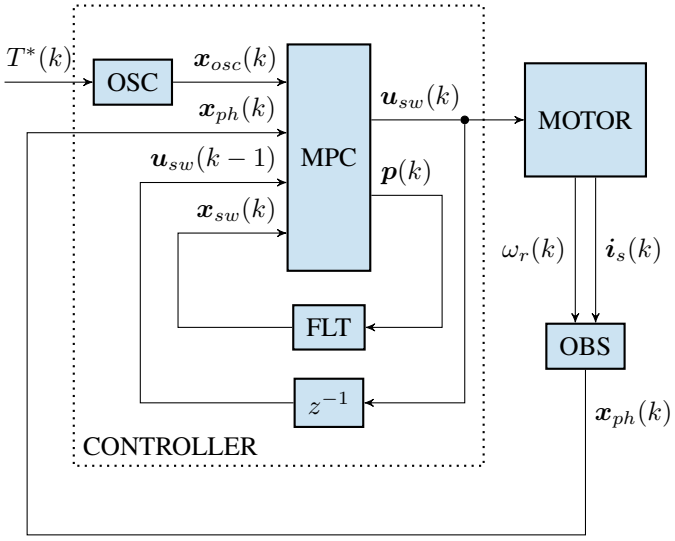


Fig. 2. Block diagram of the control loop. The controller within the dotted line receives the desired torque $T^*(k)$ and the motor states $\boldsymbol{x}_{ph}(k)$ providing the switch position $\boldsymbol{u}_{sw}(k)$.

$\boldsymbol{x}_{osc}(k), \boldsymbol{x}_{ph}(k), \boldsymbol{u}_{sw}(k-1)$ and $\boldsymbol{x}_{sw}(k)$ as in Figure 2 a new optimization problem is then solved providing an updated optimal switching sequence, and so on. The whole control algorithm, appearing within the dotted line, runs within $25\,\mu$s.

F. Approximate Dynamic Programming

The goal of this section is to compute a value function approximation V^{adp} for an infinite horizon version of (18). The function V^{adp} is used as a tail cost in (18).

Let $V^*(\boldsymbol{z})$ be the value function evaluated in $\boldsymbol{z}$, i.e. the optimal value of the objective of our control problem starting at state $\boldsymbol{z}$

$$V^*(\boldsymbol{z}) = \min_{\boldsymbol{u}\in\mathcal{U}(\boldsymbol{z})} \left\{ \sum_{k=0}^{\infty} \gamma^k l(\boldsymbol{x}(k), \boldsymbol{u}(k)) \right\},$$

subject to the system dynamics (18b). For notational convenience, we will drop the time index k from the vectors in this section. The main idea behind dynamic programming is that the function V^* is the unique solution to the equation

$$V^*(\boldsymbol{z}) = \min_{\boldsymbol{u}\in\mathcal{U}(\boldsymbol{z})} \{ l(\boldsymbol{z}, \boldsymbol{u}) + \gamma V^*(\boldsymbol{A}\boldsymbol{z} + \boldsymbol{B}\boldsymbol{u}) \} \quad \forall \boldsymbol{z},$$

known as the Bellman equation. The right-hand side can be written as monotonic operator $\mathcal{T}$ on V^*, usually referred to as the Bellman operator: $V^* = \mathcal{T}V^*$. Once V^* is known, the optimal control policy for our problem starting at state $\boldsymbol{z}$ can be found as

$$\boldsymbol{\psi}^*(\boldsymbol{z}) = \arg\min_{\boldsymbol{u}\in\mathcal{U}(\boldsymbol{z})} \{ l(\boldsymbol{z}, \boldsymbol{u}) + \gamma V^*(\boldsymbol{A}\boldsymbol{z} + \boldsymbol{B}\boldsymbol{u}) \},$$

subject to constraints (18b).

Unfortunately, solutions to the Bellman equation can only be solved analytically in a limited number of special cases; e.g. when the state and inputs have small dimensions or when the system is linear, unconstrained and the cost function is quadratic [29]. For more complicated problems, dynamic programming is limited by the so-called curse of dimensionality; storage and computation requirements tend to grow exponentially with the problem dimensions. Because of the integer switches in the power converter analyzed in this work, it is intractable to compute the optimal infinite horizon cost and policy and, hence, systematic methods for approximating the optimal value function offline are needed.

Approximate dynamic programming [19] consists of various techniques for estimating V^* using knowledge from the system dynamics, fitted data through machine learning or iterative learning through simulations.

Approximation via Iterated Bellman Inequalities: the approach developed in [20] and [21] relaxes the Bellman equation into an inequality

$$V^{adp}(\boldsymbol{z}) \leq \min_{\boldsymbol{u}\in\mathcal{U}(\boldsymbol{z})} \left\{ l(\boldsymbol{z}, \boldsymbol{u}) + \gamma V^{adp}(\boldsymbol{A}\boldsymbol{z} + \boldsymbol{B}\boldsymbol{u}) \right\}, \quad \forall \boldsymbol{z}, \tag{20}$$

or, equivalently, using the Bellman operator: $V^{adp} \leq \mathcal{T}V^{adp}$.

The set of functions V^{adp} that satisfy the Bellman inequality are underestimators of the optimal value function V^*. This happens because, if V^{adp} satisfies $V^{adp} \leq \mathcal{T}V^{adp}$, then by the monotonicity of the operator $\mathcal{T}$ and value iteration convergence, we can write

$$V^{adp} \leq \mathcal{T}V^{adp} \leq \mathcal{T}\left(\mathcal{T}V^{adp}\right) \leq \cdots \leq \lim_{i\to\infty} \mathcal{T}^i V^{adp} = V^*.$$

The Bellman inequality is therefore a sufficient condition for underestimation of V^*. In [21] the authors show that by iterating inequality (20), the conservatism of the approximation can be reduced. The iterated Bellman inequality is defined as:

$$V^{adp}(\boldsymbol{z}) \leq \mathcal{T}^M V^{adp},$$

where $M > 1$ is an integer defining the number of iterations. This inequality is equivalent to the existence of functions V_i^{adp} such that

$$V^{adp} \leq \mathcal{T}V_1^{adp}, \quad V_1^{adp} \leq \mathcal{T}V_2^{adp}, \quad \ldots V_{M-1}^{adp} \leq \mathcal{T}V^{adp}.$$

By defining $V_0^{adp} = V_M^{adp} = V^{adp}$, we can rewrite the iterated inequality as

$$V_{i-1}^{adp} \leq \mathcal{T}V_i^{adp}, \quad i = 1, \ldots, M, \tag{21}$$

where V_i^{adp} are the iterates of the value function.

To make the problem tractable, we will restrict the iterates to the finite-dimensional subspace spanned by the basis functions $V^{(j)}$ defined in [20], [21]:

$$V_i^{adp} = \sum_{j=1}^{K} \alpha_{ij} V^{(j)}, \quad i = 0, \ldots, M-1. \tag{22}$$

The coefficients α_i will be computed by solving a Semidefinite Program (SDP) [22].

The rewritten iterated Bellman inequality in (21) suggests the following optimization problem for finding the best under-

estimator for the value function V^* :

$$\begin{aligned}
\text{maximize} \quad & \int_{\mathcal{X}} V^{adp}(\boldsymbol{z}) c(\mathrm{d}\boldsymbol{z}) && \text{(23a)}\\
\text{subject to} \quad & V_{i-1}^{adp}(\boldsymbol{z}) \leq \min_{\boldsymbol{u}\in\mathcal{U}(\boldsymbol{z})} \left\{ l(\boldsymbol{z},\boldsymbol{u}) + \gamma V_i^{adp}(\boldsymbol{A}\boldsymbol{z}+\boldsymbol{B}\boldsymbol{u}) \right\} && \text{(23b)}\\
& \forall \boldsymbol{z} \in \mathbb{R}^6 \times \{-1,0,1\}, \quad i = 1,\dots,M, && \text{(23c)}\\
& V_0^{adp} = V_M^{adp} = V^{adp}, && \text{(23d)}
\end{aligned}$$

where $c(\cdot)$ is a non-negative measure over the state space. On the chosen subspace (22), the inequality (23b) is convex in the coefficients α_{ij}. To see this, note that the left-hand side is affine in α_{ij}. Moreover, for a fixed $\boldsymbol{u}$ the argument of $\min$ on the right-hand side is affine in α_{ij} while the $\min$ of affine functions is concave.

The solution to (23) is the function spanned by the chosen basis that maximizes the c-weighted 1-norm defined in the cost function while satisfying the iterated Bellman inequality [20]. Hence, $c(\cdot)$ can be regarded as a distribution giving more importance to regions of the state space where we would like a better approximation.

Following the approach in [21], we make use of quadratic candidate functions of the form

$$V_i^{adp}(\boldsymbol{z}) = \boldsymbol{z}^\top \boldsymbol{P}_i \boldsymbol{z} + 2\boldsymbol{q}_i^\top \boldsymbol{z} + r_i, \quad i = 0,\dots,M, \qquad (24)$$

where $\boldsymbol{P}_i \in \mathbb{S}^{n_x}$, $\boldsymbol{q}_i \in \mathbb{R}^{n_x}$, $r_i \in \mathbb{R}$, $i = 0,\dots,M$.

If we denote $\boldsymbol{\mu}_c \in \mathbb{R}^{n_x}$ and $\boldsymbol{\Sigma}_c \in \mathbb{S}_+^{n_x}$ as the mean and the covariance matrix of measure $c(\cdot)$ respectively, by using candidate functions as in (24) the cost function of problem (23) becomes

$$\int_{\mathcal{X}} V^{adp}(\boldsymbol{z}) c(\mathrm{d}\boldsymbol{z}) = \mathrm{Tr}\left(\boldsymbol{P}_0 \boldsymbol{\Sigma}_c\right) + 2\boldsymbol{q}_0^\top \boldsymbol{\mu}_c + r_0.$$

We now focus on rewriting the constraint (23b) as a Linear Matrix Inequality (LMI) [30]. We first remove the $\min$ on the right-hand side by imposing the constraint for every admissible $\boldsymbol{u} \in \mathcal{U}(\boldsymbol{x}_0)$ and obtain

$$\begin{aligned}
& V_{i-1}^{adp}(\boldsymbol{z}) \leq l(\boldsymbol{z},\boldsymbol{u}) + \gamma V_i^{adp}(\boldsymbol{A}\boldsymbol{z}+\boldsymbol{B}\boldsymbol{u}),\\
& \forall \boldsymbol{z} \in \mathbb{R}^6 \times \{-1,0,1\},\ \forall \boldsymbol{u} \in \mathcal{U}(\boldsymbol{z}),\ i = 1,\dots,M.
\end{aligned} \qquad (25)$$

From [21], we can rewrite (25) as a quadratic form

$$\begin{aligned}
& \begin{bmatrix} \boldsymbol{z} \\ 1 \end{bmatrix}^\top \boldsymbol{M}_i(\boldsymbol{u}) \begin{bmatrix} \boldsymbol{z} \\ 1 \end{bmatrix} \geq 0,\ \forall \boldsymbol{z} \in \mathbb{R}^6 \times \{-1,0,1\},\\
& \forall \boldsymbol{u} \in \mathcal{U}(\boldsymbol{z}),\ i = 1,\dots,M,
\end{aligned} \qquad (26)$$

where

$$\boldsymbol{M}_i(\boldsymbol{u}) = \boldsymbol{L} + \gamma \boldsymbol{G}_i(\boldsymbol{u}) - \boldsymbol{S}_{i-1} \in \mathbb{S}^{n_x}. \qquad (27)$$

is a symmetric matrix. The matrices $\boldsymbol{S}_{i-1}, \boldsymbol{L}$ and $\boldsymbol{G}_i(\boldsymbol{u})$ are defined in Appendix B.

By noting that the state vector $\boldsymbol{z}$ includes two parts which can take only a finite set of values — the normalized desired frequency fixed to 1 and the previous physical input $\boldsymbol{u}_{sw}(k-1) \in \{-1,0,1\}$ — we can explicitly enumerate part of the state-space and rewrite the quadratic form (26) more compactly as

$$\begin{bmatrix} \tilde{\boldsymbol{z}} \\ 1 \end{bmatrix}^\top \tilde{\boldsymbol{M}}_i(\boldsymbol{m}) \begin{bmatrix} \tilde{\boldsymbol{z}} \\ 1 \end{bmatrix} \geq 0,\ \forall \tilde{\boldsymbol{z}} \in \mathbb{R}^8,\ \forall \boldsymbol{m} \in \mathcal{M},\ i = 1,\dots,M, \qquad (28)$$

where $\tilde{\boldsymbol{z}}$ is the state vector without the desired frequency and $\boldsymbol{u}_{sw}(k-1)$. Moreover, $\boldsymbol{m} := (\boldsymbol{u}_{sw}, \boldsymbol{u}_{sw,pr}) \in \mathcal{M}$ are all the possible combinations of current and previous switch positions satisfying the switching and integrality constraints (19). The detailed derivation of $\tilde{\boldsymbol{M}}(\boldsymbol{m}) \in \mathbb{S}^9$ can be found in Appendix B.

Using the non-negativity condition of quadratic forms [22], it is easy to see that (28) holds if and only if $\tilde{\boldsymbol{M}}_i(\boldsymbol{m})$ is positive semidefinite. Hence, problem (23) can finally be rewritten as the following SDP

$$\begin{aligned}
\text{maximize} \quad & \mathrm{Tr}\left(\boldsymbol{P}_0 \boldsymbol{\Sigma}_c\right) + 2\boldsymbol{q}_0^\top \boldsymbol{\mu}_c + r_0\\
\text{subject to} \quad & \tilde{\boldsymbol{M}}_i(\boldsymbol{m}) \succeq 0, \quad \forall \boldsymbol{m} \in \mathcal{M}, \quad i = 1,\dots,M\\
& V_0^{adp} = V_M^{adp}\\
& \boldsymbol{P}_i \in \mathbb{S}^{n_x},\ \boldsymbol{q}_i \in \mathbb{R}^{n_x},\ r_i \in \mathbb{R},\ i = 0,\dots,M,
\end{aligned} \qquad (29)$$

which can be solved efficiently using a standard SDP solver, e.g. [31]. Once we obtain the solution to (29), we can define the infinite horizon tail cost to be used in problem (18) as

$$V^{adp}(\boldsymbol{z}) = \boldsymbol{z}^\top \boldsymbol{P}_0 \boldsymbol{z} + 2\boldsymbol{q}_0^\top \boldsymbol{z} + r_0. \qquad (30)$$

G. Optimization Problem in Vector Form

Since we consider short horizons, we adopt a condensed MPC formulation of problem (18) with only input variables, producing a purely integer program. In this way all the possible discrete input combinations can be evaluated directly. With a sparse formulation including the continuous states within the variables, it would be necessary to solve a mixed-integer program requiring more complex computations.

Let us define the input sequence over the horizon N starting at time 0 as

$$\boldsymbol{U} = \left[\boldsymbol{u}(0)^\top\ \boldsymbol{u}(1)^\top\ \dots\ \boldsymbol{u}(N-1)^\top\right]^\top, \qquad (31)$$

where we have dropped the time index from $\boldsymbol{U}$ to simplify the notation. With straightforward algebraic manipulations outlined in Appendix C, it is possible to rewrite problem (18) as a parametric integer quadratic program in the initial state $\boldsymbol{x}_0$:

$$\begin{aligned}
\text{minimize} \quad & \boldsymbol{U}^\top \boldsymbol{Q} \boldsymbol{U} + 2\boldsymbol{f}\left(\boldsymbol{x}_0\right)^\top \boldsymbol{U}\\
\text{subject to} \quad & \boldsymbol{A}_{ineq} \boldsymbol{U} \leq \boldsymbol{b}_{ineq}(\boldsymbol{x}_0)\\
& \mathcal{G}\boldsymbol{U} \in \{-1,0,1\}^{3N}.
\end{aligned} \qquad (32)$$

IV. Framework for Performance Evaluation

To benchmark our algorithm we consider a neutral point clamped voltage source inverter connected to a medium-voltage induction machine and a constant mechanical load. We consider the same model as in [13]: a 3.3 kV and 50 Hz squirrel-cage induction machine rated at 2 MVA with a total leakage inductance of 0.25 pu. On the inverter side, we assume the dc-link voltage $V_{dc} = 5.2\,\mathrm{kV}$ to be constant and the potential of the neutral point to be fixed. The base

quantities of the per unit (pu) system are the following: $V_b = \sqrt{2/3}V_{rat} = 2694\,\mathrm{V}$, $I_b = \sqrt{2}I_{rat} = 503.5\,\mathrm{A}$ and $f_b = f_{rat} = 50\,\mathrm{Hz}$. Quantities V_{rat}, I_{rat} and f_{rat} refer to the rated voltage, current and frequency respectively. The detailed parameters are provided in Table I. The switching frequency is typically in the range between 200 and 350 Hz for medium-voltage inverters [13]. If not otherwise stated, all simulations were done at rated torque, nominal speed and fundamental frequency of 50 Hz.

TABLE I
RATED VALUES AND PARAMETERS OF THE DRIVE [13]

Induction Motor				Inverter	
Voltage	3300 V	R_s	0.0108 pu	V_{dc}	1.930 pu
Current	356 A	R_r	0.0091 pu	x_c	11.769 pu
Real power	1.587 MW	X_{ls}	0.1493 pu		
Apparent power	2.035 MVA	X_{lr}	0.1104 pu		
Frequency	50 Hz	X_m	2.3489 pu		
Rotational speed	596 rpm				

We consider an idealized model with the semiconductors switching instantaneously. As such, we neglect second-order effects like deadtimes, controller delays, measurement noise, observer errors, saturation of the machine's magnetic material, variations of the parameters and so on. This is motivated by the fact that, using a similar model, previous simulations [32] showed a very close match with the experimental results in [33]. All the steady-state simulations in the following sections were also performed with model mismatch of $\pm 1\,\%$ in all the parameters of Table I showing negligible variations in the THD. However, we omit these benchmarks since an exhaustive sensitivity analysis is out of the scope of this paper.

V. Achievable Performance in Steady State

We performed closed loop simulations in steady-state operation in MATLAB to benchmark the achievable performance in terms of THD and switching frequency. The system was simulated for 4 periods before recording to ensure it reaches steady-state operation. The THD and switching frequency were computed over simulations of 20 periods. The discount factor was chosen as $\gamma = 0.95$ and the switching frequency filter parameters as $r_1 = r_2 = 800$ in order to get a smooth estimate. The weighting δ was chosen such that the switching frequency is around 300 Hz. The infinite horizon estimation SDP (29) is formulated using YALMIP [34] with $M = 50$ Bellman iterations and solved offline using MOSEK [31]. Note that in case of a change in the systems parameters, e.g. the dc-link voltage or the rotor speed, the tail cost has to be recomputed. However, it possible to precompute offline and store several quadratic tail costs for different possible parameters and evaluate the desired one online without significant increase complexity.

For comparison, we simulated the drive system also with the direct MPC controller described in [13] (denoted DMPC) tuned in order to have the same switching frequency by adjusting the weighting parameter λ_u.

The integer optimization problems were solved using Gurobi Optimizer [35]. Numerical results with both approaches are presented in Table II. Note that the choice of the solver does not influence the THD or the switching frequency and we would have obtained the same results with another optimization software.

TABLE II
SIMULATION RESULTS WITH ADP AND WITH DMPC FROM [14] AT SWITCHING FREQUENCY 300 Hz

	ADP		DMPC [14]	
	δ	THD [%]	λ_u	THD [%]
$N = 1$	4	5.24	0.00235	5.44
$N = 2$	5.1	5.13	0.00690	5.43
$N = 3$	5.5	5.10	0.01350	5.39
$N = 10$	10	4.80	0.10200	5.29

Our method, with a horizon of $N = 1$ provided both a THD improvement over the DMPC formulation in [13] with $N = 10$ and a drastically better numerical speed. This showed how choosing a meaningful cost function can provide good control performance without recourse to long horizons. Moreover, we also performed a comparison with longer horizons $N = 2$, $N = 3$ and $N = 10$. Our method, with horizon $N = 10$ would give an even greater reduction in THD to $4.80\,\%$.

VI. FPGA Implementation

A. Hardware Setup

We implemented the control algorithm on a Xilinx Zynq (xc7z020) [36], a low cost FPGA, running at approximately 150 MHz mounted on the Zedboard evaluation module [37]; see Figure 3. The control algorithm was coded in C++ using

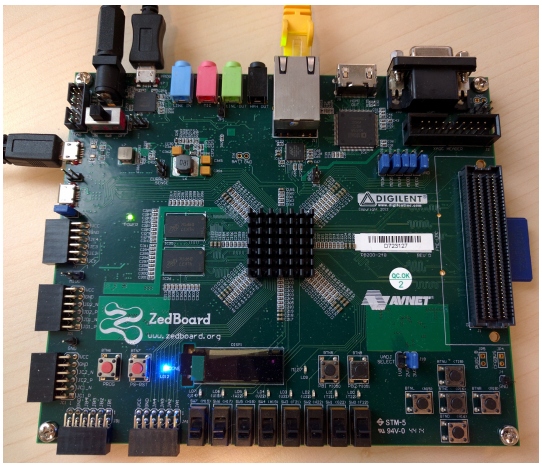


Fig. 3. Zedboard Evaluation Board used for HIL Tests. The controller runs on the FPGA while the plant is simulate on the laptop. The states and input vectors are passed via the ethernet cable (yellow). The micro-usb cable on the left side provides a UART interface with the laptop used to print if there are any problems in the communication. The cable in the top left corner is connected to the power supply while the other micro-usb cable next to it provides access to the USB-JTAG interface to program the FPGA module.

the PROTOIP Toolbox [38]. The FPGA vendors HLS tool

Xilinx Vivado HLS [39] was used to convert the written code to VHDL defining the Programmable Logic connections.

B. Algorithm Description

We now present a detailed description of how the controller within the dotted lines in Figure 2 was implemented on the FPGA.

The updates in OSC and FLT were implemented as simple matrix multiplications. The solver for the integer problem (32) was implemented with a simple exhaustive search algorithm for three reasons: first, the tail cost approximation provides good performance with very few horizon steps while considering a relatively small number of input combinations; second, the structure of the problem allows us to evaluate both the inequalities and cost function for multiple input sequences in parallel; third, the FPGA logic is particularly suited for highly pipelined and/or parallelized operations, which are at the core of exhaustive search.

To exploit the FPGA architecture, we implemented our algorithm in fixed-point arithmetic using custom data types defined in Vivado HLS [39]. In particular, we used 4 integer and 0 fractional bits to describe the integer inputs and 2 integer and 22 fractional bits to describe the states and the cost function values. This choice is given by the minimum number of bits necessary to describe these quantities from floating-point simulations in Section V. Note that the exhaustive search algorithm does not suffer from any accumulation of rounding error because it consists entirely of independent function evaluations, in contrast to iterative optimization algorithms [3].

We provide pseudo-code for our method in Algorithm 1. From Figure 2, the controller receives the required torque $T^*(k)$ and the motor states $\boldsymbol{x}_{ph}(k)$ and returns the switch positions $\boldsymbol{u}_{sw}(k)$. From line 2 to line 8 the oscillator OSC and the filter FLT are updated to compute the new initial state $\boldsymbol{x}_0$ for the optimization algorithm. Note that if there is a change in the required torque then the oscillator states $\boldsymbol{x}_{osc}(k)$ are reset to match the new $T^*(k)$. Line 9 and 10 precompute the vectors in problem (32) depending on $\boldsymbol{x}_0$.

The main loop iterating over all input combinations is split into two subloops: Loop 1, which is completely decoupled and can be parallelized; and Loop 2, which can only be pipelined.

Loop 1 from line 11 to 19 computes the cost function values for every combination i and stores it into vector $\boldsymbol{J}$. All the possible input sequence combinations are saved in the static matrix $\boldsymbol{U}^{seq}$. For every loop cycle, sequence i is saved into variable $\boldsymbol{u}$. Then, in line 13, the value of $\boldsymbol{p}(k)$ is updated inside $\boldsymbol{u}$ with $\boldsymbol{u}_{sw}(k-1)$ according to (12) and (13). If $\boldsymbol{u}$ satisfies the constraint $\boldsymbol{A}_{ineq}\boldsymbol{u} \leq \boldsymbol{b}_{ineq}(\boldsymbol{x}_0)$, then the cost function is stored in $\boldsymbol{J}_{(i)}$ (line 15). Otherwise $\boldsymbol{J}_{(i)}$ is set to a high value J_{ub}. Note that, even though it would bring considerable speed improvements, we do not precompute offline the quadratic part $\boldsymbol{u}^\top \boldsymbol{Q}\boldsymbol{u}$ of the cost and the left side of the inequality $\boldsymbol{A}_{ineq}\boldsymbol{u}$ since it would also require enumeration over inputs at the previous control cycle used in line 13.

Each iteration of this loop is independent from the others and can therefore be parallelized efficiently.

Loop 2 from line 20 to 26 is a simple loop iterating over the computed cost function values to find the minimum and save it into J_{min}. Every iteration depends sequentially on J_{min} which is accessed and can be modified at every i. Thus, in this form it is not possible to parallelize this loop, although it can be pipelined.

Algorithm 1 Controller Algorithm

1: **function** COMPUTEMPCINPUT($T^*(k), \boldsymbol{x}_{ph}(k)$)
Data: $\boldsymbol{x}_{osc}(k-1)$, $\boldsymbol{x}_{sw}(k-1)$, $\boldsymbol{p}(k-1)$ and $\boldsymbol{u}_{sw}(k-1)$
Parameters: $\boldsymbol{U}^{seq} \in \mathbb{Z}^{6\times 27^N}$, $J_{ub} \in \mathbb{R}$
Initialize: $\boldsymbol{J} \in \mathbb{R}^{27^N}$, $J_{min} \in \mathbb{R}$ and $i_{min} \in \mathbb{N}$

Execute Filter and Oscillator to Obtain Initial State:
2: **if** change in $T(k)$ **then**
3: $\boldsymbol{x}_{osc}(k) \leftarrow$ Reset according to (3)
4: **else**
5: $\boldsymbol{x}_{osc}(k) \leftarrow \boldsymbol{A}_{osc}\boldsymbol{x}_{osc}(k-1)$
6: **end if**
7: $\boldsymbol{x}_{sw}(k) \leftarrow \boldsymbol{A}_{sw}\boldsymbol{x}_{sw}(k-1) + \boldsymbol{B}_{sw}\boldsymbol{p}(k-1)$
8: $\boldsymbol{x}_0 \leftarrow \left[\boldsymbol{x}_{ph}(k)^\top \ \boldsymbol{x}_{osc}(k)^\top \ \boldsymbol{x}_{sw}(k)^\top \ \boldsymbol{u}_{sw}(k-1)^\top\right]^\top$

Precompute Vectors:
9: $\boldsymbol{f}(\boldsymbol{x}_0) \leftarrow$ Compute from (40)
10: $\boldsymbol{b}_{ineq}(\boldsymbol{x}_0) \leftarrow$ Compute from (41), (42)

Loop 1 - Compute Cost Function Values:
11: **for** $i = 1, \ldots, 27^N$ **do**
12: $\boldsymbol{u} \leftarrow \boldsymbol{U}^{seq}_{(:,i)}$
13: $\boldsymbol{u}_{(4:6)} \leftarrow \boldsymbol{p}(k) = \|\boldsymbol{u}_{(1:3)} - \boldsymbol{u}_{sw}(k-1)\|_1$
14: **if** $\boldsymbol{A}_{ineq}\boldsymbol{u} \leq \boldsymbol{b}_{ineq}(\boldsymbol{x}_0)$ **then**
15: $\boldsymbol{J}_{(i)} \leftarrow \boldsymbol{u}^\top \boldsymbol{Q}\boldsymbol{u} + 2\boldsymbol{f}(\boldsymbol{x}_0)^\top \boldsymbol{u}$
16: **else**
17: $\boldsymbol{J}_{(i)} \leftarrow J_{ub}$
18: **end if**
19: **end for**

Loop 2 - Find Minimum:
20: $J_{min} \leftarrow J_{ub}$, $i_{min} \leftarrow 1$
21: **for** $i = 1, \ldots, 27^N$ **do**
22: **if** $\boldsymbol{J}_{(i)} \leq J_{min}$ **then**
23: $J_{min} \leftarrow \boldsymbol{J}_{(i)}$
24: $i_{min} \leftarrow i$
25: **end if**
26: **end for**

Return Results
27: $\boldsymbol{u}_{sw}(k) \leftarrow \boldsymbol{U}^{seq}_{(1:3,i_{min})}$ and $\boldsymbol{p}(k) \leftarrow \boldsymbol{U}^{seq}_{(4:6,i_{min})}$
28: **return** $\boldsymbol{u}_{sw}(k)$
29: **end function**

C. Circuit Generation

In Vivado HLS [39] it is possible to specify directives to optimize the circuit synthesis according to the resources available on the target board. Loop 1 and Loop 2 were pipelined and the preprocessing operations from line 2 to 10 parallelized. We generated the circuit for the algorithm 1 with horizons $N = 1$ and $N = 2$ at frequency 150 MHz (clock

cycle of 7 ns). The resources usage and the timing estimates are displayed in Table III. Since timing constraints were met, there was no need to parallelize Loop 1 to reduce computation time.

TABLE III
RESOURCES USAGE AND TIMING ESTIMATES FOR IMPLEMENTATION ON THE XILINX ZYNQ FPGA (XC7Z020) RUNNING AT 150 MHz

		$N=1$	$N=2$
FPGA Resources	LUT	15127 (28 %)	31028 (58 %)
	FF	11156 (10 %)	20263 (19 %)
	BRAM	6 (2 %)	21 (7 %)
	DSP	89 (40 %)	201 (91 %)
Clock Cycles		371	1953
Delay		2.60 μs	13.67 μs

Note that for $N = 2$ we are using already $91\,\%$ of the DSP multipliers. This is due to the limited amount of resources available on our chosen low-cost hardware.

VII. HARDWARE IN THE LOOP TESTS

We performed hardware in the loop (HIL) experiments using the controller FPGA fixed-point implementation developed in Section VI and the machine model described in Section IV.

The control loop was operated using the PROTOIP toolbox [38]: the plant model was simulated on a Macbook Pro 2.8 GHz Intel Core i7 with 16GB of RAM while the control algorithm was entirely executed on the Zedboard development board described in Section VI-A.

A. Steady State

The controller was benchmarked in HIL in steady-state operation to compare its performance to the achievable performance results obtained in Table II. We chose the same controller parameters as in Section V.

The HIL tests for horizon $N = 1$ are shown in Figure 4 in the per unit system. The three-phase stator currents are displayed over a fundamental period in Figure 4a, the three spectra are shown in Figure 4b with THD of $5.23\,\%$ and the input sequences are plotted in Figure 4c.

From the experimental benchmarks with horizon $N = 1$ and $N = 2$ we obtained $\mathrm{THD} = 5.23\,\%$ and $\mathrm{THD} = 5.14\,\%$ respectively. As expected, these results are very close to the simulated ones in Table II. The slight difference ($\sim 0.01\,\%$) comes from the fixed-point implementation of the oscillator OSC and the filter FLT in Figure 2.

B. Transients

One of the main advantages of direct MPC is the fast transient response [13]. We tested torque transients in HIL with the same tuning parameters as in the steady state benchmarks. At nominal speed, reference torque steps were imposed; see Figure 6b. These steps were translated into different current references to track, as shown in Figure 6a, while the computed inputs are shown in Figure 6c.

The torque step from 1 to 0 in the per unit system presented an extremely short settling time of 0.35 ms similar to deadbeat control approaches [40]. This was achieved by inverting the voltage applied to the load. Since we prohibited switchings between -1 and 1 in (19b) and (19a), the voltage inversion was performed in $2T_s$ via an intermediate zero switching position.

Switching from 0 to 1 torque produced much slower response time of approximately 3.5 ms. This was due to the limited available voltage in the three-phase admissible switching positions. As shown in Figure 6c, during the second step at time 20 ms, the phases b and c saturated at the values $+1$ and -1 respectively for the majority of the transient providing the maximum available voltage that could steer the currents to the desired values.

These results match the simulations of the DMPC formulation in [13] in terms of settling time showing that our method possesses the fast dynamical behavior during transients typical of direct current MPC.

As noted in [13], having a longer horizon or a better predictive behavior does not significantly improve the settling times. This is because the benefit of longer prediction obtainable by extending the horizon or adopting a powerful final stage costs is reduced by the saturation of the inputs during the transients.

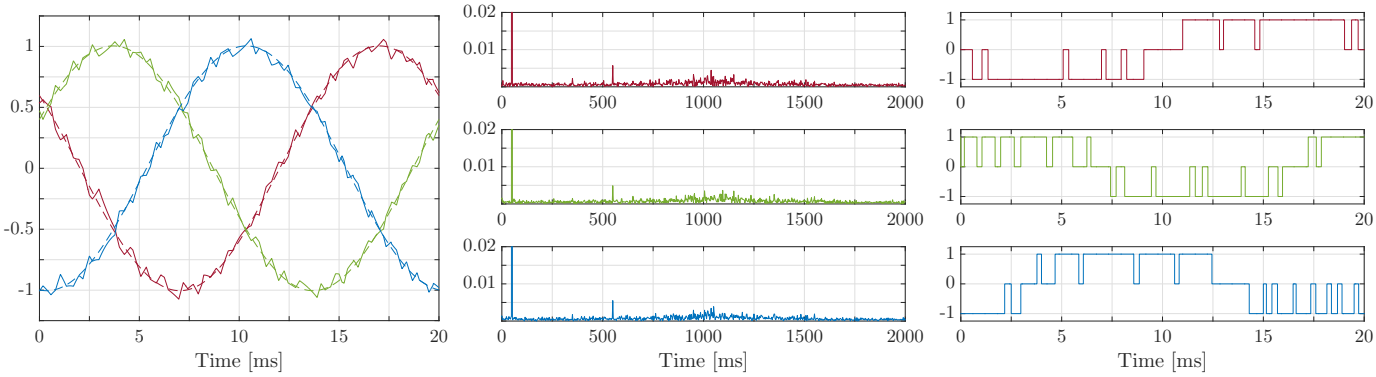
C. Execution Time

To show that the controller is able to run on cheap hardware within $T_s = 25\,\mu$s, we measured the time the FPGA took to execute Algorithm 1 for horizon $N = 1$ and $N = 2$. Since there are no available DMPC sphere decoding algorithm execution times, we compared our results to the time needed to solve the DMPC formulation in [14] for the same horizon lengths on a Macbook Pro with Intel Core i7 2.8 GHz and 16GB of RAM using the commercial integer program solver Gurobi Optimizer [35] which implements an efficient Branch-and-Bound algorithm. The results are shown in Figure 5.

The FPGA execution times were $5.76\,\mu$s and $17.27\,\mu$s for horizon $N = 1$ and $N = 2$ respectively. Note that they presented a slight overhead of approximately $3.5\,\mu$s compared to the estimates in Table III since the measured times included the time needed to exchange the input-output data from the FPGA to the ARM processor through the RAM memory. Without the overhead, the estimated FPGA computing times obtained by the circuit generation are exact; see [39].

Note that the time needed by the FPGA to compute the control algorithm is deterministic with zero variance. This makes our HIL implementation particularly suited for real-time applications. Furthermore, it is important to underline that the method we propose is the *only* method available that can produce integer optimal solutions to this problem achieving this performance in $25\,\mu$s sampling time.

The execution times needed by Gurobi optimizer were $621.2 \pm 119.98\,\mu$s and $750.40 \pm 216.15\,\mu$s for horizons $N = 1$ and $N = 2$ respectively. The non-negligible standard deviation appeared because of the branch-and-bound algorithm implemented in Gurobi. However, since we are considering real-time applications, we are interested in the worst case number of visited nodes which is always the whole tree of



(a) Three-phase stator currents (solid lines) with their references (dashed lines).

(b) Stator current spectrum.

(c) Three-phase switch positions inputs.

Fig. 4. Waveforms produced during HIL Tests by the direct model predictive controller at steady state operation, at full speed and rated torque. Horizon of $N = 1$ is used. The switching frequency is approximately 300 Hz and the current THD is 5.23 %.

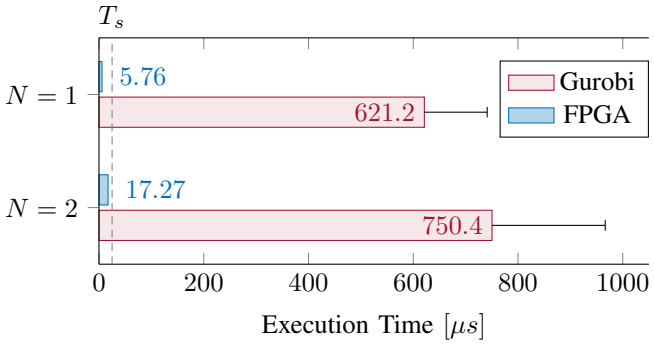


Fig. 5. Execution times required by the Xilinx Zynq FPGA (xc7z020) to execute our controller based on an ADP formulation (blue) and using Gurobi Optimizer [35] to solve the formulation in [13] on a Macbook Pro with Intel Core i7 2.8 GHz and 16GB of RAM

combinations, i.e. 27^N. Note that the DMPC formulation was solved in [13] using a different branch-and-bound algorithm based on the sphere decoding algorithm [15], but the worst case number of visited nodes cannot be easily reduced because of the $\mathcal{NP}-$hardness of the problem.

VIII. Conclusion and Future Work

This work proposes a new computationally efficient direct model predictive control (MPC) scheme for current reference tracking in power converters. We extended the problem formulation in [13] and [14] in order to include a switching frequency estimator in the system state and rewrite the optimal control problem as a regulation one. To reduce the horizon length and decrease the computational burden while preserving good control performances, we estimated the infinite horizon tail cost of the MPC problem formulation using approximate dynamic programming (ADP).

Steady-state simulation results show that with our method requiring short horizons, it is possible to obtain better performance than the direct MPC formulation in [13] with long horizons. This is due to the predictive behavior of the tail cost function obtained with ADP.

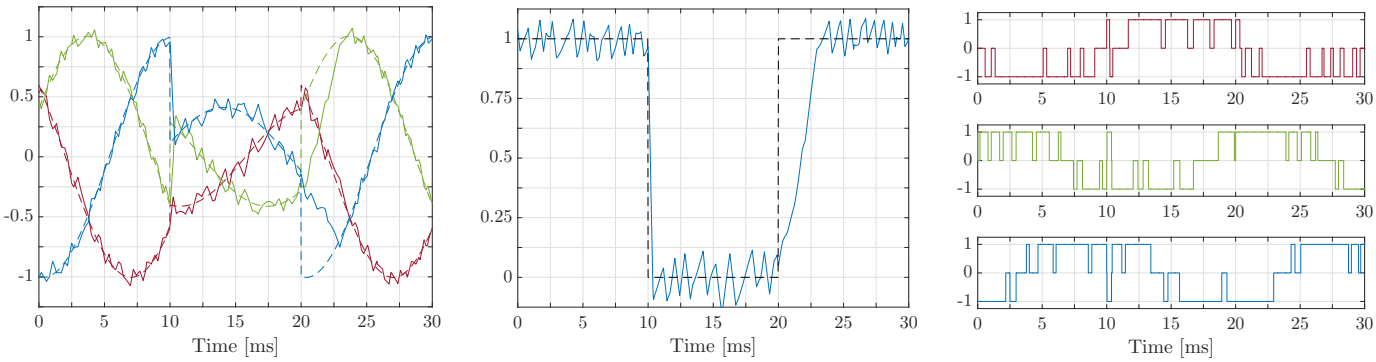
The control algorithm was implemented in fixed-point arithmetic on the low size Xilinx Zynq FPGA (xc7z020) for horizons $N = 1$ and $N = 2$. Hardware in the loop (HIL) tests during steady-state operation showed an almost identical performance to the simulation results. We also performed transient simulations where our proposed approach exhibited the same very fast dynamic response as the direct MPC described in [13]. Moreover, we showed that our algorithm can run within the sampling time of 25 μs by measuring the execution time on the FPGA. Results showed that only 5.76 μs and 17.27 μs are required to run our controller for horizons $N = 1$ and $N = 2$ respectively.

Direct MPC can also be applied to more complex schemes such as modular multilevel converters (MMC) [41]. While it is possible to derive a complete MMC model that could be used in an MPC approach, the number of switching levels per horizon stage exponentially increases with the number of converter levels. As stated in [14], long-horizon predictive power is expected to be even more beneficial with MMCs. We believe that our method, making use of short prediction horizons and long predictions using an approximate value function could be applied effectively to MMCs with more levels because it is still possible to evaluate on commercially available FPGAs the multilevel feasible switching combinations over very short horizons within the required sampling time.

There are several future directions to be investigated. Given the system design there are several symmetries in the model that could be exploited to increase the controller horizon without requiring more computational power. Regarding the frequency estimation, other filters with different orders could be implemented and their parameters chosen optimally by solving an optimization problem instead of performing manual tuning. Moreover, it would be interesting to benchmark other ADP tail cost functions (e.g. higher order polynomials) to understand which ones best approximate the infinite horizon tail cost and produce the best overall control performance.

Acknowledgment

The authors would like to thank Andrea Suardi and Bulat Khusainov for their advice regarding FPGA implementation,



(a) Three-phase stator currents (solid lines) with their references (dashed lines).

(b) Actual (solid line) and reference (dashed line) torques.

(c) Three-phase switch positions inputs.

Fig. 6. Reference torque steps produced by the direct model predictive controller in HIL tests with horizon $N = 1$.

in particular the PROTOIP toolbox.

Appendix A
Physical System Matrices

The matrices corresponding to the continuous-time physical system in (4) are

$$\boldsymbol{D} = \begin{bmatrix} -\frac{1}{\tau_s} & 0 & \frac{X_m}{\tau_r D} & \omega_r \frac{X_m}{D} \\ 0 & -\frac{1}{\tau_s} & -\omega_r \frac{X_m}{D} & \frac{X_m}{\tau_r D} \\ \frac{X_m}{\tau_r} & 0 & -\frac{1}{\tau_r} & -\omega_r \\ 0 & \frac{X_m}{\tau_r} & \omega_r & -\frac{1}{\tau_r} \end{bmatrix}$$

$$\boldsymbol{E} = \frac{X_r}{D}\frac{V_{dc}}{2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \boldsymbol{P}, \qquad \boldsymbol{F} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

Appendix B
ADP Formulation

The matrices defining the quadratic form are

$$\boldsymbol{S}_{i-1} \coloneqq \begin{bmatrix} \boldsymbol{P}_{i-1} & \boldsymbol{q}_{i-1} \\ \boldsymbol{q}_{i-1}^\top & r_{i-1} \end{bmatrix}, \quad \boldsymbol{L} \coloneqq \begin{bmatrix} \boldsymbol{C}^\top \boldsymbol{C} & \boldsymbol{0}_{n_x \times 1} \\ \boldsymbol{0}_{n_x \times 1}^\top & 0 \end{bmatrix}$$

$$\boldsymbol{G}_i(\boldsymbol{u}) \coloneqq \begin{bmatrix} \boldsymbol{\Psi}^{(i)} & \boldsymbol{\Phi}^{(i)}(\boldsymbol{u}) \\ \boldsymbol{\Phi}^{(i)}(\boldsymbol{u})^\top & \Gamma^{(i)}(\boldsymbol{u}) \end{bmatrix},$$

with

$$\begin{aligned} \boldsymbol{\Psi}^{(i)} &= \boldsymbol{A}^\top \boldsymbol{P}_{i-1} \boldsymbol{A} \\ \boldsymbol{\Phi}^{(i)}(\boldsymbol{u}) &= \boldsymbol{A}^\top \boldsymbol{P}_i \boldsymbol{B}\boldsymbol{u} + \boldsymbol{A}^\top \boldsymbol{q}_i \\ \Gamma^{(i)}(\boldsymbol{u}) &= \boldsymbol{u}^\top \boldsymbol{B}^\top \boldsymbol{P}_i \boldsymbol{B}\boldsymbol{u} + 2\boldsymbol{q}_i^\top \boldsymbol{B}\boldsymbol{u} + r_i, \end{aligned}$$

for $i = 1, \ldots, M$.

The quadratic form decomposition can be derived as follows. For every $\boldsymbol{m} = (\boldsymbol{u}_{sw}, \boldsymbol{u}_{sw,pr}) \in \mathcal{M}$, we can define the input

$$\boldsymbol{u}_{\boldsymbol{m}} = \begin{bmatrix} \boldsymbol{u}_{sw} \\ \|\boldsymbol{u}_{sw} - \boldsymbol{u}_{sw,pr}\|_1 \end{bmatrix},$$

and the matrix $\boldsymbol{M}_i(\boldsymbol{u}_{\boldsymbol{m}})$ using (27). Now we can decompose the vector in quadratic form (26) using the state definition (17)

$$\begin{bmatrix} \boldsymbol{z}^\top & 1 \end{bmatrix}^\top = \begin{bmatrix} \boldsymbol{z}_{ph}^\top & \boldsymbol{z}_{osc}^\top & \boldsymbol{z}_{sw,1:2}^\top & \boldsymbol{z}_{sw,3}^\top & \boldsymbol{z}_{upr}^\top & 1 \end{bmatrix}^\top.$$

The matrix $\boldsymbol{M}_i(\boldsymbol{u}_{\boldsymbol{m}})$ can also be decomposed in the same fashion into smaller block matrices as follows

$$\begin{bmatrix} \boldsymbol{z}_{ph} \\ \boldsymbol{z}_{osc} \\ \boldsymbol{z}_{sw,1:2} \\ \boldsymbol{z}_{sw,3} \\ \boldsymbol{z}_{upr} \\ 1 \end{bmatrix}^\top \left[\begin{array}{c|ccc} \boldsymbol{M}_{i,11} & \boldsymbol{M}_{i,12} & \boldsymbol{M}_{i,13} & \boldsymbol{M}_{i,14} \\ \hline \boldsymbol{M}_{i,12}^\top & \boldsymbol{M}_{i,22} & \boldsymbol{M}_{i,23} & \boldsymbol{M}_{i,24} \\ \boldsymbol{M}_{i,13}^\top & \boldsymbol{M}_{i,23}^\top & \boldsymbol{M}_{i,33} & \boldsymbol{M}_{i,34} \\ \boldsymbol{M}_{i,14}^\top & \boldsymbol{M}_{i,24}^\top & \boldsymbol{M}_{i,34}^\top & \boldsymbol{M}_{i,44} \end{array}\right] \begin{bmatrix} \boldsymbol{z}_{ph} \\ \boldsymbol{z}_{osc} \\ \boldsymbol{z}_{sw,1:2} \\ \boldsymbol{z}_{sw,3} \\ \boldsymbol{z}_{upr} \\ 1 \end{bmatrix} \geq 0,$$

where the dependency $\boldsymbol{M}_{i,**}(\boldsymbol{u}_m)$, $\boldsymbol{m} \in \mathcal{M}$ has been neglected to simplify the notation. The first row and first column block matrices have the first and second dimensions respectively equal to the length of vector $\begin{bmatrix} \boldsymbol{z}_{ph}^\top & \boldsymbol{z}_{osc}^\top & \boldsymbol{z}_{sw,1:2}^\top \end{bmatrix}^\top$. Since $\boldsymbol{z}_{upr} = \boldsymbol{u}_{sw,pr}$ and $\boldsymbol{z}_{sw,3} = 1$ (normalized desired switching frequency), we can rewrite the quadratic form as

$$\begin{bmatrix} \boldsymbol{z}_{ph} \\ \boldsymbol{z}_{osc} \\ \boldsymbol{z}_{sw,1:2} \\ 1 \end{bmatrix}^\top \left[\begin{array}{c|c} \boldsymbol{M}_{i,11} & \boldsymbol{\Psi}_{i,1} \\ \hline \boldsymbol{\Psi}_{i,1}^\top & \boldsymbol{\Psi}_{i,2} \end{array}\right] \begin{bmatrix} \boldsymbol{z}_{ph} \\ \boldsymbol{z}_{osc} \\ \boldsymbol{z}_{sw,1:2} \\ 1 \end{bmatrix} \geq 0, \qquad (33)$$

where

$$\begin{aligned} \boldsymbol{\Psi}_{i,1} &= \boldsymbol{M}_{i,13}\boldsymbol{z}_{upr} + \boldsymbol{M}_{i,12} + \boldsymbol{M}_{i,14} \\ \boldsymbol{\Psi}_{i,2} &= \boldsymbol{z}_{upr}^\top \boldsymbol{M}_{i,33}\boldsymbol{z}_{upr} + 2\boldsymbol{M}_{i,23}\boldsymbol{z}_{upr} + 2\boldsymbol{M}_{i,34}^\top \boldsymbol{z}_{upr} \\ &\quad + 2\boldsymbol{M}_{i,24} + \boldsymbol{M}_{i,22} \end{aligned}$$

Therefore, we will denote the matrix in (33) as $\tilde{\boldsymbol{M}}_i(\boldsymbol{m})$ and the quadratic form vectors as $\begin{bmatrix} \tilde{\boldsymbol{z}}^\top & 1 \end{bmatrix}^\top$.

Appendix C
Dense Formulation of the MPC Problem

By considering the input sequence (31) and the state sequence over the horizon denoted as

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}(0)^\top & \boldsymbol{x}(1)^\top & \ldots & \boldsymbol{x}(N)^\top \end{bmatrix}^\top,$$

the system dynamics (18b) with initial state constraint (18c) can be written as

$$\boldsymbol{X} = \mathcal{A}\boldsymbol{x}_0 + \mathcal{B}\boldsymbol{U}, \tag{34}$$

where $\mathcal{A}$ and $\mathcal{B}$ are

$$\mathcal{A} \coloneqq \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{A} \\ \vdots \\ \boldsymbol{A}^N \end{bmatrix}, \quad \mathcal{B} \coloneqq \begin{bmatrix} \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{B} & & \vdots \\ \boldsymbol{AB} & \boldsymbol{B} & \\ \vdots & \ddots & \ddots & \boldsymbol{0} \\ \boldsymbol{A}^{N-1}\boldsymbol{B} & \cdots & \boldsymbol{AB} & \boldsymbol{B} \end{bmatrix}.$$

Let us separate cost function (18a) in two parts: the cost from stage 0 to $N-1$ and the tail cost. The former can be rewritten as

$$\begin{aligned} &\sum_{k=0}^{N-1} \gamma^k \|\boldsymbol{C}\boldsymbol{x}(k)\|_2^2 = \boldsymbol{X}^\top \mathcal{H} \boldsymbol{X} \\ &= \boldsymbol{U}^\top \mathcal{B}^\top \mathcal{H} \mathcal{B} \boldsymbol{U} + 2\left(\mathcal{B}^\top \mathcal{H} \mathcal{A} \boldsymbol{x}_0\right)^\top \boldsymbol{U} + \text{const}(\boldsymbol{x}_0), \end{aligned} \tag{35}$$

where the last equality is obtained by plugging in (34) and the term $\text{const}(\boldsymbol{x}_0)$ is a constant depending on the initial state. Matrix $\mathcal{H}$ is defined as

$$\mathcal{H} = \begin{bmatrix} \boldsymbol{C}^\top\boldsymbol{C} & & & \boldsymbol{0} \\ & \gamma\boldsymbol{C}^\top\boldsymbol{C} & & \\ & & \ddots & \vdots \\ & & & \gamma^{N-1}\boldsymbol{C}^\top\boldsymbol{C} \\ \boldsymbol{0} & & \cdots & \boldsymbol{0} \end{bmatrix}. \tag{36}$$

In order derive the tail cost, let us write the last stage as

$$\boldsymbol{x}(N) = \boldsymbol{A}^N \boldsymbol{x}_0 + \mathcal{B}_{end}\boldsymbol{U}, \tag{37}$$

where $\mathcal{B}_{end}$ is the last row of $\mathcal{B}$ used to compute the last state. Using equations (37) and (30), the tail cost can be rewritten as

$$\begin{aligned} &V^{adp}(\boldsymbol{x}(N)) = \boldsymbol{x}(N)^\top \boldsymbol{P}_0 \boldsymbol{x}(N) + 2\boldsymbol{q}_0^\top \boldsymbol{x}(N) + r_0 \\ &= \boldsymbol{U}^\top \left(\mathcal{B}_{end}^\top \boldsymbol{P}_0 \boldsymbol{B}_{end}\right) \boldsymbol{U} + 2\left(\mathcal{B}_{end}^\top \boldsymbol{P}_0 \boldsymbol{A}^N \boldsymbol{x}_0 + \mathcal{B}_{end}^\top \boldsymbol{q}_0\right)^\top \boldsymbol{U} \\ &\quad + \text{const}(\boldsymbol{x}_0). \end{aligned} \tag{38}$$

By combining (35) and (38) according to (18a), we obtain the full cost function

$$\boldsymbol{J} = \boldsymbol{U}^\top \boldsymbol{Q} \boldsymbol{U} + 2\boldsymbol{f}\left(\boldsymbol{x}_0\right)^\top \boldsymbol{U} + \text{const}(\boldsymbol{x}_0),$$

with

$$\boldsymbol{Q} = \mathcal{B}^\top \mathcal{H} \mathcal{B} + \gamma^N \mathcal{B}_{end}^\top \boldsymbol{P}_0 \mathcal{B}_{end} \tag{39}$$

$$\begin{aligned} \boldsymbol{f}\left(\boldsymbol{x}_0\right) = &\left(\mathcal{B}^\top \mathcal{H} \mathcal{A} + \gamma^N \mathcal{B}_{end} \boldsymbol{P}_0 \mathcal{A}^N\right) \boldsymbol{x}_0 \\ &+ \gamma^N \mathcal{B}_{end}^\top \boldsymbol{q}_0. \end{aligned} \tag{40}$$

We now rewrite the constraints of problem (18) in vector form. Inequalities (19b) with $k = 0, \ldots, N-1$ can be written as

$$\mathcal{R}\boldsymbol{U} \leq \mathcal{S}\boldsymbol{X} \iff (\mathcal{R} - \mathcal{S}\mathcal{B})\boldsymbol{U} \leq \mathcal{S}\mathcal{A}\boldsymbol{x}_0, \tag{41}$$

where in the term on the right we substituted (34).

Similarly, constraint (19a) with $k = 0, \ldots, N-1$ can be written as

$$\left\|\boldsymbol{T}\boldsymbol{u}(k)\right\|_\infty \leq 1 \iff \mathcal{F}\boldsymbol{U} \leq \boldsymbol{1}, \tag{42}$$

where $\boldsymbol{1}$ is a vector of ones of appropriate dimensions. Matrices $\mathcal{R}, \mathcal{S}$ and $\mathcal{F}$ are

$$\mathcal{R} = \begin{bmatrix} \boldsymbol{I}-\boldsymbol{T} & & \\ & \ddots & \\ & & \boldsymbol{I}-\boldsymbol{T} \\ -\boldsymbol{I}-\boldsymbol{T} & & \\ & \ddots & \\ & & -\boldsymbol{I}-\boldsymbol{T} \end{bmatrix},$$

$$\mathcal{S} = \begin{bmatrix} \boldsymbol{W} & & \boldsymbol{0} \\ & \ddots & \\ & \boldsymbol{W} & \vdots \\ -\boldsymbol{W} & & \\ & \ddots & \\ & -\boldsymbol{W} & \boldsymbol{0} \end{bmatrix}, \; \mathcal{F} = \begin{bmatrix} \boldsymbol{T} & & \\ & \ddots & \\ & & \boldsymbol{T} \\ -\boldsymbol{T} & & \\ & \ddots & \\ & & -\boldsymbol{T} \end{bmatrix}.$$

Let us define matrix $\mathcal{G}$ extrapolating all the switch positions $\boldsymbol{u}_{sw}(k)$ from $\boldsymbol{U}$ as

$$\mathcal{G} = \begin{bmatrix} G & & \\ & \ddots & \\ & & G \end{bmatrix}.$$

We can now merge (41) and (42) into a single inequality $\boldsymbol{A}_{ineq}\boldsymbol{U} \leq \boldsymbol{b}_{ineq}(\boldsymbol{x}_0)$ and rewrite (18) neglecting the constant terms in the cost function obtaining the result in (32).

REFERENCES

[1] J. B. Rawlings and D. Q. Mayne, *Model Predictive Control: Theory and Design* . Nob Hill Publishing, LLC, 2014.

[2] P. Cortes, M. P. Kazmierkowski, R. M. Kennel, D. E. Quevedo, and J. Rodriguez, "Predictive Control in Power Electronics and Drives," *IEEE Transactions on Industrial Electronics*, vol. 55, no. 12, pp. 4312–4324, Nov. 2008.

[3] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. Springer Science & Business Media, 2006.

[4] J. L. Jerez, P. J. Goulart, S. Richter, G. A. Constantinides, E. C. Kerrigan, and M. Morari, "Embedded Online Optimization for Model Predictive Control at Megahertz Rates," *IEEE Transactions on Automatic Control*, vol. 59, no. 12, pp. 3238–3251, Nov. 2014.

[5] R. Ghaemi, J. Sun, and I. V. Kolmanovsky, "An integrated perturbation analysis and Sequential Quadratic Programming approach for Model Predictive Control," *Automatica*, vol. 45, no. 10, pp. 2412–2418, Oct. 2009.

[6] Y. Xie, R. Ghaemi, J. Sun, and J. S. Freudenberg, "Model Predictive control for a Full Bridge DC/DC Converter," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 1, pp. 164–172, Jan. 2012.

[7] T. Geyer, "Low Complexity Model Predictive Control in Power Electronics and Power Systems," Ph.D. dissertation, ETH Zürich, 2005.

[8] D. Bertsimas and R. Weismantel, *Optimization over integers*. Belmont, Massachussetts: Dynamic Ideas, 2005.

[9] D. E. Quevedo, G. C. Goodwin, and J. A. De Doná, "Finite constraint set receding horizon quadratic control," *International Journal of Robust and Nonlinear Control*, 2004.

[10] M. Morari and J. H. Lee, "Model predictive control: past, present and future," *Computers & Chemical Engineering*, vol. 23, no. 4-5, pp. 667–682, 1999.

[11] H. Nademi and L. E. Norum, "Implicit Finite Control Set Model Predictive Current Control for Modular Multilevel Converter Based on IPA-SQP Algorithm," in *In Proc. 31st Annual IEEE Applied Power Electronics Conference and Exposition (APEC 2016 Conf.)*, Long Beach, California, USA, 2016, pp. 3291–3296.

[12] P. Karamanakos, T. Geyer, N. Oikonomou, F. D. Kieferndorf, and S. Manias, "Direct Model Predictive Control: A Review of Strategies That Achieve Long Prediction Intervals for Power Electronics," *IEEE Industrial Electronics Magazine*, vol. 8, no. 1, pp. 32–43, Mar. 2014.
[13] T. Geyer and D. E. Quevedo, "Multistep Finite Control Set Model Predictive Control for Power Electronics," *IEEE Transactions on Power Electronics*, vol. 29, no. 12, pp. 6836–6846, 2014.
[14] ——, "Performance of Multistep Finite Control Set Model Predictive Control for Power Electronics," *IEEE Transactions on Power Electronics*, vol. 30, no. 3, pp. 1633–1644, 2015.
[15] B. Hassibi and H. Vikalo, "On the sphere-decoding algorithm I. Expected complexity," *IEEE Transactions on Signal Processing*, vol. 53, no. 8, pp. 2806–2818, Jul. 2005.
[16] P. Karamanakos, T. Geyer, and R. Kennel, "Reformulation of the long-horizon direct model predictive control problem to reduce the computational effort," in *Energy Conversion Congress and Exposition (ECCE)*. IEEE, 2014, pp. 3512–3519.
[17] ——, "Suboptimal search strategies with bounded computational complexity to solve long-horizon direct model predictive control problems," in *Energy Conversion Congress and Exposition (ECCE), 2015 IEEE*. IEEE, 2015, pp. 334–341.
[18] S. Kouro, P. Cortes, R. Vargas, U. Ammann, and J. Rodriguez, "Model Predictive Control - A Simple and Powerful Method to Control Power Converters," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 6, pp. 1826–1838, May 2009.
[19] D. P. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific Belmont, Massachusetts, 1996.
[20] D. P. de Farias and B. Van Roy, "The linear programming approach to approximate dynamic programming," *Operations Research*, 2003.
[21] Y. Wang, B. O'Donoghue, and S. Boyd, "Approximate dynamic programming via iterated Bellman inequalities," *International Journal of Robust and Nonlinear Control*, 2014.
[22] L. Vandenberghe and S. Boyd, "Semidefinite Programming," *SIAM Review*, vol. 38, no. 1, pp. 49–95, 1996.
[23] P. Beuchat, A. Georghiou, and J. Lygeros, "Approximate Dynamic Programming: a Q-Function Approach," *ArXiv e-prints*, Feb. 2016.
[24] P. C. Krause, O. Wasynczuk, and S. D. Sudhoff, *Analysis of Electric Machinery and Drive Systems*, 2nd ed. Hoboken, NJ, USA: Wiley, 2002.
[25] J. Holtz, "The representation of AC machine dynamics by complex signal flow graphs," *IEEE Transactions on Industrial Electronics*, vol. 42, no. 3, pp. 263–271, 1995.
[26] J. Scoltock, T. Geyer, and U. K. Madawala, "A Model Predictive Direct Current Control Strategy With Predictive References for MV Grid-Connected Converters With LCL-Filters," *IEEE Transactions on Power Electronics*, vol. 30, no. 10, pp. 5926–5937, 2015.
[27] W. C. Duesterhoeft, M. W. Schulz, and E. Clarke, "Determination of Instantaneous Currents and Voltages by Means of Alpha, Beta, and Zero Components," *American Institute of Electrical Engineers, Transactions of the*, vol. 70, no. 2, pp. 1248–1255, Jul. 1951.
[28] A. V. Oppenheim and A. S. Willsky, *Signals and Systems*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1997.
[29] R. E. Kalman, "When Is a Linear Control System Optimal?" *Journal of Basic Engineering*, vol. 86, no. 1, pp. 51–60, 1964.
[30] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*. Philadelphia: Society for industrial and applied mathematics, 1994, vol. 15.
[31] MOSEK ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 7.1 (Revision 35).*, 2015.
[32] T. Geyer, G. Papafotiou, and M. Morari, "Model Predictive Direct Torque Control - Part I: Concept, Algorithm, and Analysis," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 6, pp. 1894–1905, May 2009.
[33] G. Papafotiou, J. Kley, K. G. Papadopoulos, P. Bohren, and M. Morari, "Model Predictive Direct Torque Control - Part II: Implementation and Experimental Evaluation," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 6, pp. 1906–1915, May 2009.
[34] J. Löfberg, "YALMIP : A Toolbox for Modeling and Optimization in MATLAB," in *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.
[35] Gurobi Optimization, Inc., "Gurobi Optimizer Reference Manual," Tech. Rep., 2015.
[36] Xilinx, Inc., *Zynq-7000 All Programmable SoC Technical Reference Manual*.
[37] Avnet Inc., *Zedboard Hardware User's Guide*, 2nd ed.
[38] A. Suardi and E. C. Kerrigan, "Fast FPGA prototyping toolbox for embedded optimization," *European Control Conference (ECC)*, pp. 2589–2594, 2015.
[39] Xilinx, Inc., *Vivado Design Suite User Guide - High-Level Synthesis*.
[40] J. Rodriguez and P. Cortes, *Predictive control of power converters and electrical drives*. John Wiley & Sons, 2012, vol. 40.
[41] B. S. Riar, T. Geyer, and U. K. Madawala, "Model Predictive Direct Current Control of Modular Multilevel Converters: Modeling, Analysis, and Experimental Evaluation," *IEEE Transactions on Power Electronics*, vol. 30, no. 1, pp. 431–439, Aug. 2014.

Bartolomeo Stellato (S'16) received the B.Sc. degree in automation engineering from Politecnico di Milano, Milano, Italy in 2012 and the M.Sc. degree in robotics, systems and control from ETH Zürich, Zürich, Switzerland in 2014. He is pursuing his D.Phil. (Ph.D.) degree in control systems and optimization at the University of Oxford under the supervision of Professor Paul Goulart.

His current research interests are integer optimal control of fast dynamical systems, model predictive control and discrete optimization.

Tobias Geyer (M'08-SM'10) received the Dipl.-Ing. and Ph.D. degrees in electrical engineering from ETH Zürich, Zürich, Switzerland, in 2000 and 2005, respectively.

From 2006 to 2008, he was with the High Power Electronics Group of GE's Global Research Centre, Munich, Germany, where he focused on control and modulation schemes for large electrical drives. Subsequently, he spent three years at the Department of Electrical and Computer Engineering, The University of Auckland, Auckland, New Zealand, where he developed model predictive control schemes for medium-voltage drives. In 2012, he joined ABB's Corporate Research Centre, Baden-Dattwil, Switzerland. His research interests include model predictive control, medium-voltage drives and utility-scale power converters. He has authored and co-authored about 100 peer-reviewed publications, 25 patent applications and the book "Model predictive control of high power converters and industrial drives".

Dr. Geyer was a recipient of the 2014 Third Best Paper Award of the Transactions on Industry Applications. He also received two Prize Paper Awards at conferences. From 2011 until 2014 he served as an Associate Editor of the Industrial Drives Committee for the Transactions on Industry Applications. Since 2013 he has been serving as an Associate Editor for the Transactions on Power Electronics.

Paul J. Goulart (M'04) received the B.Sc. and M.Sc. degrees in aeronautics and astronautics from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA. He was selected as a Gates Scholar at the University of Cambridge, Cambridge, U.K., where he received the Ph.D. degree in control engineering in 2007.

From 2007 to 2011, he was a Lecturer in control systems in the Department of Aeronautics, Imperial College London, and from 2011 to 2014 a Senior Researcher in the Automatic Control Laboratory, ETH Zürich. He is currently an Associate Professor in the Department of Engineering Science and Tutorial Fellow, St. Edmund Hall, University of Oxford, Oxford, U.K. His research interests include model predictive control, robust optimization, and control of fluid flows.