Computationally Efficient Long-Horizon Direct Model Predictive Control for Transient Operation

Petros Karamanakos, *Member, IEEE*, Tobias Geyer, *Senior Member, IEEE*,
and Ricardo P. Aguilera, *Member, IEEE*

***Abstract*—In this paper we present modifications in the sphere decoder initially introduced in [1] and modified in [2] that allow for its implementation in transient operation. By investigating the geometry of the integer problem underlying direct model predictive control (MPC), a new sphere that guarantees feasibility and includes a significant smaller number of candidate solutions is computed. In a first analysis, the computational complexity can be reduced by up to 99.7% when a variable speed drive system consisting of a three-level neutral point clamped (NPC) voltage source inverter and a medium-voltage induction machine is examined. As also shown, optimality is sacrificed only to a limited extent, thus maintaining the very fast transient response inherent to direct MPC.**

I. Introduction

Over the past decade model predictive control (MPC) [3] has paved its way in becoming one of the most attractive control alternatives in power electronics [4], [5]. MPC is formulated as a constrained optimization problem, meaning that it can operate the system at its physical limits, and—as a result—achieve the best possible performance. Moreover, it can easily handle systems with nonlinear dynamics, and/or multiple inputs and outputs since it is formulated in the (discrete) time domain. These features of MPC, combined with the vast computational power readily available, justify its widespread acceptance from the power electronic community.

The most utilized MPC strategy in power electronics is the so-called finite control set MPC (FCS-MPC) [4]. The output reference tracking and the modulation problem are formulated in one computational stage, i.e., akin to a direct control strategy. Although this approach seems to be straightforward and intuitive, from a mathematical optimization perspective, it is computationally very challenging. The underlying optimization problem is a (mixed) integer program, which is known to be NP-hard, i.e., it can become computationally intractable as the number of decision variables (i.e., control inputs) or the length of the prediction horizon increase. Given that the latter has been documented to be necessary for an improved system performance [6]–[9], the commonly used brute-force method of exhaustive enumeration—according to which all candidate solutions are enumerated to determine the solution—cannot be considered as a realistic option for horizons longer than one step.

To allow for longer horizons, several techniques have been proposed that either utilize nontrivial prediction horizons [10] by implementing concepts such as move blocking [11] (e.g., [12]) and extrapolation [5] (e.g., [13]), or branch-and-bound methods [14] based on smart heuristics [6]. With regards to the latter, a dedicated branch-and-bound algorithm, developed in the field of communications and referred to as sphere decoder [15], [16], has been recently introduced in the field of power electronics [1], [2], [7]. Although it has been shown to be reasonably effective, since it manages to significantly reduce the computational complexity of the problem at steady-state operation as well as to tellingly deal with state/output constraints [17], there exist challenges that have not yet been fully addressed.

Among the most prominent issues yet to be resolved is the system operation during transient phenomena. Besides some preliminary investigations for a time-invariant system [18], a more in-depth analysis is required. As explained in this paper, because of the geometry of the problem under transient conditions, the sphere decoder tends to be less effective, rendering the algorithm computationally prohibitively demanding. To overcome this issue, this paper proposes refinements in the sphere decoding algorithm that—despite the fact they sacrifice optimality (but only marginally, as shown)—alleviate its computational burden. To highlight the effectiveness of the modified sphere decoder, a variable speed drive system, consisting of a three-level neutral point clamped (NPC) voltage source inverter driving a medium-voltage (MV) induction machine (IM), is chosen as a case study. As it is shown, the computational complexity of the algorithm—in terms of real-time floating point operations (flops)—can be reduced by up to 99.7% when long horizons, such as ten steps, are considered.

II. Optimal Control Problem

The problem examined relates to the control of the stator current of an IM when driven by a three-level NPC voltage source inverter with a constant dc-link voltage V_{dc} and a fixed neutral point potential (Fig. 1).

In the sequel, the mathematical model of the examined system and the formulation of the optimization problem are presented. Both these tasks are performed in the stationary orthogonal $\alpha\beta$ system. Therefore, any variable in the abc-plane $\boldsymbol{\xi}_{abc} = [\xi_a \ \xi_b \ \xi_c]^T$ is mapped into a two-dimensional

P. Karamanakos is with the Faculty of Computing and Electrical Engineering, Tampere University of Technology, 33101 Tampere, Finland; e-mail: p.karamanakos@ieee.org

T. Geyer is with ABB Corporate Research, 5405 Baden-Dättwil, Switzerland; e-mail: t.geyer@ieee.org

R. P. Aguilera is with the School of Electrical, Mechanical and Mechatronic Systems, University of Technology Sydney, Sydney, NSW 2007, Australia; e-mail: raguilera@ieee.org

978-1-5090-2998-3/17/$31.00 ©2017 IEEE

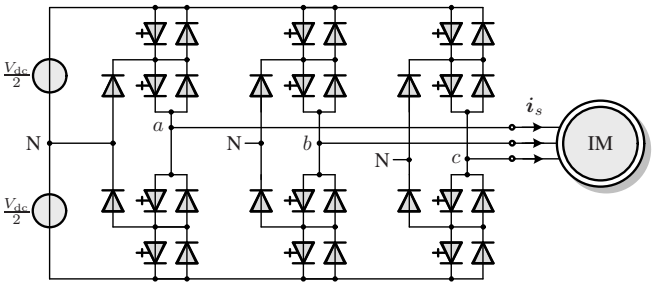


Fig. 1: Three-level three-phase neutral point clamped (NPC) voltage source inverter driving an induction motor (IM) with a fixed neutral point potential.

vector $\boldsymbol{\xi}_{\alpha\beta} = [\xi_\alpha \ \xi_\beta]^T$ in the $\alpha\beta$-plane via the transformation matrix $\boldsymbol{K}$, i.e., $\boldsymbol{\xi}_{\alpha\beta} = \boldsymbol{K}\boldsymbol{\xi}_{abc}$, with

$$\boldsymbol{K} = \frac{2}{3}\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix}.$$

A. Control Model

The output potential of each phase of the three-level NPC inverter can obtain three discrete voltage levels $-\frac{V_{\rm dc}}{2}$, 0, $\frac{V_{\rm dc}}{2}$, depending on the position of the power switches in the respective phase [19]. As a result, the inverter output voltage—equal to the voltage applied to the machine terminals $\boldsymbol{v}_{s,\alpha\beta}$—is[1]

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

where $\boldsymbol{u} = [u_a \ u_b \ u_c]^T \in \boldsymbol{\mathcal{U}} = \mathcal{U}\times\mathcal{U}\times\mathcal{U} = \mathcal{U}^3$ is the three-phase switch position, with the integer variable $u_x \in \mathcal{U} = \{-1, 0, 1\}$ denoting the switch position in phase $x \in \{a, b, c\}$.

With regards to the squirrel-cage IM, the stator current $\boldsymbol{i}_{s,\alpha\beta}$ and the rotor flux $\boldsymbol{\psi}_{r,\alpha\beta}$ in the $\alpha\beta$-plane as well as the angular speed of the rotor ω_r are adopted as state variables to fully describe its dynamics. Given the model parameters, i.e., the stator R_s and rotor R_r resistances, the stator X_{ls}, rotor X_{lr} and mutual X_m reactances, the moment of inertia H, and the mechanical load torque T_ℓ, the differential equations that govern the machine are[2] [20]

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

$$\frac{\mathrm{d}\boldsymbol{\psi}_r}{\mathrm{d}t} = \frac{X_m}{\tau_r}\boldsymbol{i}_s - \frac{1}{\tau_r}\boldsymbol{\psi}_r + \omega_r\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\boldsymbol{\psi}_r \tag{2b}$$

$$\frac{\mathrm{d}\omega_r}{\mathrm{d}t} = \frac{1}{H}(T_e - T_\ell)\,. \tag{2c}$$

In (2), the stator time constant is $\tau_s = X_r\Phi/(R_sX_r^2 + R_rX_m^2)$ and the rotor time constant $\tau_r = X_r/R_r$. Moreover, $\Phi = X_sX_r - X_m^2$ is a constant with $X_s = X_{ls} + X_m$ and $X_r = X_{lr} + X_m$, and T_e is the electromagnetic torque. Finally, $\boldsymbol{I}$ is the identity matrix of appropriate dimension (here two-dimensional).

[1]Throughout the paper, vectors in the $\alpha\beta$-plane are denoted with the corresponding subscript. For vectors in the *abc*-plane the subscript is omitted.

[2]To simplify the notation in (2) the subscript $\alpha\beta$ is dropped from the vectors $\boldsymbol{i}_s$, $\boldsymbol{\psi}_r$ and $\boldsymbol{v}_s$.

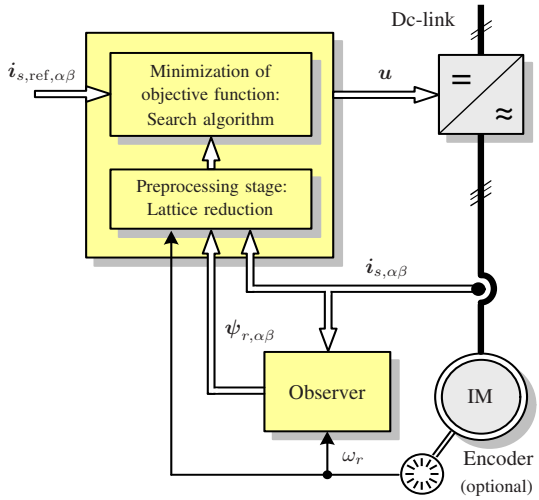


Fig. 2: Model predictive current control with reference tracking for the three-phase three-level NPC inverter with an induction machine.

For the MPC algorithm developed in Section II-B the internal model of the drive system is required so that its future behavior can be predicted. To this end, the stator current and the rotor flux in the $\alpha\beta$-plane constitute the stator vector $\boldsymbol{x} = [i_{s\alpha} \ i_{s\beta} \ \psi_{r\alpha} \ \psi_{r\beta}]^T$. Note that due to the slower dynamics the mechanical speed is assumed to be constant within the prediction horizon. Hence, the rotor angular speed ω_r is not a state variable, but a time-varying parameter instead. The input to the system is considered to be the three-phase switch position $\boldsymbol{u}$, whereas the stator current is the output variable, i.e., $\boldsymbol{y} = \boldsymbol{i}_{s,\alpha\beta}$. Therefore, by taking into account (1) and (2), the continuous-time state-space representation of the drive can be written as

$$\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} = \boldsymbol{D}\boldsymbol{x}(t) + \boldsymbol{E}\boldsymbol{K}\boldsymbol{u}(t) \tag{3a}$$

$$\boldsymbol{y}(t) = \boldsymbol{F}\boldsymbol{x}(t)\,, \tag{3b}$$

where the dynamics $\boldsymbol{D}$, input $\boldsymbol{E}$ and output $\boldsymbol{F}$ matrices, are given in Appendix A.

Using exact Euler discretization, the discrete-time state-space model of the drive is

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{K}\boldsymbol{u}(k) \tag{4a}$$

$$\boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k)\,, \tag{4b}$$

with $\boldsymbol{A} = \mathbf{e}^{\boldsymbol{D}T_s}$, $\boldsymbol{B} = -\boldsymbol{D}^{-1}(\boldsymbol{I} - \boldsymbol{A})\boldsymbol{E}$ and $\boldsymbol{C} = \boldsymbol{F}$. Moreover, $\boldsymbol{I}$ here is the 4×4 identity matrix, $\mathbf{e}$ the matrix exponential, T_s the sampling interval, and $k \in \mathbb{N}$.

B. Direct MPC With Current Reference Tracking

The discussed control algorithm aims to regulate the stator current $\boldsymbol{i}_{s,\alpha\beta}$ along its reference $\boldsymbol{i}_{s,\mathrm{ref},\alpha\beta}$, while operating the drive at low switching frequency. The latter objective is fairly important when MV drives are of concern, since it is directly

related to the switching power losses which have to be kept low for an increased efficiency of the converter. As can be seen in Fig. 2, the optimal controller meets both objectives by computing and applying the control signals (i.e., the switch positions) in one stage, i.e., a modulation stage is bypassed.

By introducing the current (i.e., output) tracking error term $\boldsymbol{i}_{s,\mathrm{err},\alpha\beta}(k) = \boldsymbol{i}_{s,\mathrm{ref},\alpha\beta}(k) - \boldsymbol{i}_{s,\alpha\beta}(k)$, and the switching (i.e., control) effort term $\Delta\boldsymbol{u}(k) = \boldsymbol{u}(k) - \boldsymbol{u}(k-1)$, the aforementioned control objectives are mapped into a scalar, i.e.,

$$J(k) = \sum_{\ell=k}^{k+N-1} ||\boldsymbol{i}_{s,\mathrm{err},\alpha\beta}(\ell+1|k)||_2^2 + \lambda_u ||\Delta\boldsymbol{u}(\ell|k)||_2^2 \,. \quad (5)$$

Based on the state of the current error and switching effort at step k, function (5) penalizes their evolution over the finite prediction horizon of length N time steps. Note that the weighting factor $\lambda_u > 0$ is introduced to adjust the trade-off between the two terms, i.e., the trade-off between the current tracking ability of the controller and the switching frequency.

To find the optimal sequence of control actions $\boldsymbol{U}^*(k) = [\boldsymbol{u}^{*T}(k)\ \boldsymbol{u}^{*T}(k+1)\ \ldots\ \boldsymbol{u}^{*T}(k+N-1)]^T$ that results in the most desirable system behavior, i.e., the one that minimizes (5), problem

$$\begin{aligned} \underset{\boldsymbol{U}(k)}{\text{minimize}} \quad & J(k) \\ \text{subject to} \quad & \boldsymbol{x}(\ell+1) = \boldsymbol{A}\boldsymbol{x}(\ell) + \boldsymbol{B}\boldsymbol{u}(\ell) \\ & \boldsymbol{y}(\ell) = \boldsymbol{C}\boldsymbol{x}(\ell)\,,\ \forall\, \ell = k, \ldots, k+N-1 \\ & \boldsymbol{U}(k) \in \mathbb{U} \end{aligned} \quad (6)$$

is solved in real time. In (6), $\boldsymbol{U}(k) = [\boldsymbol{u}^T(k)\ \boldsymbol{u}^T(k+1)\ \ldots\ \boldsymbol{u}^T(k+N-1)]^T$ is the optimization variable and $\mathbb{U} = \boldsymbol{\mathcal{U}}^N \subset \mathbb{Z}^n$, with $n = 3N$, is the feasible set defined as the N-times Cartesian product of the set $\boldsymbol{\mathcal{U}}$.

III. Equivalent Integer Least-Squares Problem

By relaxing the feasible set in (6), from $\mathbb{U}$ to $\mathbb{R}^n$, one can easily compute the so-called unconstrained solution $\bar{\boldsymbol{U}}_{\mathrm{unc}}(k) = \boldsymbol{H}\boldsymbol{U}_{\mathrm{unc}}(k) \in \mathbb{R}^n$, with $\boldsymbol{U}_{\mathrm{unc}} \in \mathbb{R}^n$ [1]. Based on $\bar{\boldsymbol{U}}_{\mathrm{unc}}$—and after some algebraic manipulations shown in Appendix B—problem (6) becomes

$$\begin{aligned} \underset{\boldsymbol{U}(k)}{\text{minimize}} \quad & ||\bar{\boldsymbol{U}}_{\mathrm{unc}}(k) - \boldsymbol{H}\boldsymbol{U}(k)||_2^2 \\ \text{subject to} \quad & \boldsymbol{U}(k) \in \mathbb{U}\,, \end{aligned} \quad (7)$$

which is an integer least-squares (ILS) problem [1]. The non-singular, upper triangular matrix $\boldsymbol{H} \in \mathbb{R}^{n\times n}$ in (7), is known as the lattice generator matrix that generates the discrete space wherein the solution lies.

ILS problems are known to be NP-hard [21], meaning that if $\boldsymbol{H}$ is ill-conditioned, then solving (7) may become computationally intractable. To avoid this, algorithms can be used that reshape the search space (i.e., they modify the entries of $\boldsymbol{H}$) with the eventual goal of having a lattice as close to orthogonal as possible, while ensuring that the length of the basis vectors that generate it (i.e., the Euclidean norm of the columns of the lattice generator matrix) is relatively small. To this end, the Lenstra-Lenstra-Lovász (LLL) lattice basis reduction algorithm [22] is employed to generate the reduced lattice generator matrix $\widetilde{\boldsymbol{H}}$. The resulting equivalent ILS problem is of the form [2]

$$\begin{aligned} \underset{\boldsymbol{U}(k)}{\text{minimize}} \quad & ||\,\widetilde{\boldsymbol{U}}_{\mathrm{unc}}(k) - \widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{U}}(k)||_2^2\,, \\ \text{subject to} \quad & \boldsymbol{U}(k) \in \mathbb{U}\,, \end{aligned} \quad (8)$$

with $\widetilde{\boldsymbol{U}}_{\mathrm{unc}}(k) = \widetilde{\boldsymbol{H}}\ \boldsymbol{M}^{-1}\boldsymbol{U}_{\mathrm{unc}}(k)$, $\widetilde{\boldsymbol{U}}(k) = \boldsymbol{M}^{-1}\boldsymbol{U}(k)$ and $\widetilde{\boldsymbol{H}} = \boldsymbol{V}^T\boldsymbol{H}\boldsymbol{M}$. Finally, matrix $\boldsymbol{V} \in \mathbb{R}^{n\times n}$ is orthogonal and matrix $\boldsymbol{M} \in \mathbb{Z}^{n\times n}$ unimodular (i.e., $\det \boldsymbol{M} = \pm 1$).

Interpreting (8), one has to find the (integer) n-dimensional lattice point that is closest (in Euclidean sense) to the (real-valued) unconstrained solution $\widetilde{\boldsymbol{U}}_{\mathrm{unc}}(k)$. Equivalently, one has to find the smallest hypersphere (n-dimensional sphere) of radius ρ centered at $\widetilde{\boldsymbol{U}}_{\mathrm{unc}}(k)$ that includes at most one lattice point. To this end, the branch-and-bound method known as sphere decoder [15] can be utilized. With sphere decoding the search space is limited to those lattice points (i.e., nodes) included in the computed hypersphere, meaning that only these points are considered as candidate solutions, and thus evaluated, while optimality is still guaranteed. As a result, this search algorithm proves to be highly computationally efficient, especially compared with the brute-force approach of the exhaustive enumeration, see [7] and [2].

As can be understood, the initial radius ρ of the computed hypersphere determines the efficacy of the sphere decoder. The *initial* sphere should be small enough to include as few nodes as possible, whereas it should be nonempty in order to avoid feasibility issues. According to [23], the initial radius can be set equal to the minimum of two options, i.e.,

$$\rho(k) = \min\{\rho_a(k), \rho_b(k)\}\,, \quad (9)$$

with the options being

$$\rho_a(k) = ||\,\widetilde{\boldsymbol{U}}_{\mathrm{unc}}(k) - \widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{U}}_{\mathrm{bab}}(k)||_2\,, \quad (10a)$$

$$\rho_b(k) = ||\,\widetilde{\boldsymbol{U}}_{\mathrm{unc}}(k) - \widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{U}}_{\mathrm{ed}}(k)||_2\,. \quad (10b)$$

The radii in (10) are computed based on different estimated solutions (i.e., initial guesses) $\widetilde{\boldsymbol{U}}_{\mathrm{est}} = \boldsymbol{M}^{-1}\boldsymbol{U}_{\mathrm{est}}$. Radius ρ_a is computed based on the so-called Babai estimate [24], i.e., $\boldsymbol{U}_{\mathrm{est}} = \boldsymbol{U}_{\mathrm{bab}}$, which results by rounding the unconstrained solution to the closest integer vector, i.e., $\boldsymbol{U}_{\mathrm{bab}}(k) = \lfloor \boldsymbol{U}_{\mathrm{unc}}(k) \rceil$. The second option (see (10b)) is computed based on an educational guess ($\boldsymbol{U}_{\mathrm{est}} = \boldsymbol{U}_{\mathrm{ed}}$), which results by taking into account the previously applied solution $\boldsymbol{U}^*(k-1)$ shifted by one time step and with a repetition of the last switch position (see (40) in [1]). Note that $\widetilde{\boldsymbol{U}}_{\mathrm{bab}}(k) = \boldsymbol{M}^{-1}\boldsymbol{U}_{\mathrm{bab}}(k)$ and $\widetilde{\boldsymbol{U}}_{\mathrm{ed}}(k) = \boldsymbol{M}^{-1}\boldsymbol{U}_{\mathrm{ed}}(k)$.

With an initial tight sphere of radius (9) the vast majority of the lattice points are a priori excluded from being considered candidate solutions. The task of the sphere decoder is to visit the remaining nodes enclosed in the sphere in order to conclude to the optimal solution. To do so, and as can be seen

in Algorithm 1 (lines 15–28), the optimizer traverses a search tree of n levels by visiting one-by-one the m-dimensional nodes, $m = 1, \ldots, n$, in a depth-first search manner. Starting form the root (top), n-dimensional node, the algorithm goes through the branches—that constitute the candidate elements of the solution—with a goal to reach the bottom level of the tree (i.e., the leaf nodes that are one-dimensional) as quickly as possible. As long as the assembled candidate solution results in a smaller intermediate sphere than the initial one, the algorithm keeps visiting nodes at the lower levels of tree; if not, it backtracks to visit higher-level nodes and find a new path to the bottom level. When a leaf node is reached, a complete candidate solution is assembled, and a new, tighter sphere is computed. The optimizer continues searching for different sequences that may result in a tighter sphere until it gets a "certificate" that the optimal solution has been found.

IV. Sphere Decoder for Transient Operation

As mentioned before—and as shown in [7] and [2]—the sphere decoder is significantly more computationally efficient than the exhaustive enumeration when the steady-state operation of the system is examined. Even though the sphere decoding principle is also applicable to transient operation, the computationally complexity of the sphere decoder itself increases (that of the exhaustive enumeration remains the same since all candidate solutions need to be *always* evaluated). The reason for that is that the unconstrained solution under transient operating conditions can be far from the (truncated) lattice, implying that the initial radius (9) can be large with a considerable number of nodes included in the resulting sphere. More specifically, radius ρ_b (see (10)) is clearly not a good guess since the previously applied solution (i.e., the one applied before the transient occurs) is expected to be very different compared to the one computed when the transient phenomenon begins, whereas ρ_a—although a better choice—appears to be a poor initial radius as well.

To overcome this, and to keep the complexity of the sphere decoding algorithm low, one could project the unconstrained solution $\widetilde{\boldsymbol{U}}_{\rm unc}(k)$ onto the convex hull of the lattice in the space generated by $\widetilde{\boldsymbol{H}}$. According to [25], th convex hull is defined as

$$\begin{aligned}\mathbf{conv}\,\widetilde{\mathbb{U}} = \{&\theta_1(\widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{U}}_1) + \cdots + \theta_j(\widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{U}}_j) \mid \widetilde{\boldsymbol{U}}_i = \boldsymbol{M}^{-1}\boldsymbol{U}_i,\\ &\boldsymbol{U}_i \in \mathbb{U},\, \theta_i \geq 0,\, i = 1, \ldots, j,\, j = 3^n,\\ &\theta_1 + \cdots + \theta_j = 1\}\,.\end{aligned} \tag{11}$$

To this end, the following quadratic program (QP) needs to be solved in real time

$$\begin{aligned}\underset{\widetilde{\boldsymbol{U}}(k)}{\text{minimize}} \quad & ||\,\widetilde{\boldsymbol{U}}_{\rm unc}(k) - \widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{U}}(k)||_2^2\\ \text{subject to} \quad & \widetilde{\boldsymbol{U}}(k) \in \mathbf{conv}\,\widetilde{\mathbb{U}}\,,\end{aligned} \tag{12}$$

the optimal value of which is the (Euclidean) distance $\mathbf{dist}(\widetilde{\boldsymbol{U}}_{\rm unc}(k), \mathbf{conv}\,\widetilde{\mathbb{U}})$, and the optimal point is the projection of $\widetilde{\boldsymbol{U}}_{\rm unc}(k)$ on $\mathbf{conv}\,\widetilde{\mathbb{U}}$ [25].

Having found the (real-valued) projected point $\widetilde{\boldsymbol{U}}_{\rm rlx}(k) \in \mathbf{conv}\,\widetilde{\mathbb{U}}$ a new sphere can be used by the sphere decoder. This smaller sphere is centered at $\widehat{\boldsymbol{U}}_{\rm rlx}$ and has radius $\hat{\rho}$ that is given by (9), with the difference that the first option results by rounding $\boldsymbol{U}_{\rm rlx}(k) = \boldsymbol{M}\,\widetilde{\boldsymbol{H}}^{-1}\widetilde{\boldsymbol{U}}_{\rm rlx}(k)$ to the nearest integer point, i.e., similarly to (10a) with the difference that $\lfloor \boldsymbol{U}_{\rm rlx}(k) \rceil$ is the estimated solution.

The downside of this approach is that the shape of the convex hull $\mathbf{conv}\,\widetilde{\mathbb{U}}$ is nontrivial (it is an n-dimensional polyhedron) that depends on $\widetilde{\boldsymbol{H}}$. Since $\widetilde{\boldsymbol{H}}$ depends on the rotor speed (see Appendix B), the convex hull needs to be recomputed when this parameter changes. This increases the computational burden of the approach rendering its benefits less tangible.

A workaround is to solve the equivalent problem of (12) in the original space, as motivated by the following remark.

Remark 1: The objective function $||\,\widetilde{\boldsymbol{U}}_{\rm unc}(k) - \widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{U}}(k)||_2^2$ can be written as $(\boldsymbol{U}_{\rm unc}(k) - \boldsymbol{U}(k))^T\boldsymbol{Q}(\boldsymbol{U}_{\rm unc}(k) - \boldsymbol{U}(k))$ with $\boldsymbol{Q} = \boldsymbol{H}^T\boldsymbol{H}$.

Proof: According to the definitions in Section III, it is true that

$$\widetilde{\boldsymbol{U}}_{\rm unc}(k) = \widetilde{\boldsymbol{H}}\,\boldsymbol{M}^{-1}\boldsymbol{U}_{\rm unc}(k)\,, \tag{13}$$

$$\widetilde{\boldsymbol{U}}(k) = \boldsymbol{M}^{-1}\boldsymbol{U}(k)\,, \tag{14}$$

$$\widetilde{\boldsymbol{H}} = \boldsymbol{V}^T\boldsymbol{H}\boldsymbol{M}\,. \tag{15}$$

Then for the function $J(k) = ||\,\widetilde{\boldsymbol{U}}_{\rm unc}(k) - \widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{U}}(k)||_2^2$, it holds that

$$\begin{aligned}J(k) &= ||\,\widetilde{\boldsymbol{U}}_{\rm unc}(k) - \widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{U}}(k)||_2^2\\ &= ||\boldsymbol{V}^T\boldsymbol{H}\boldsymbol{M}\boldsymbol{M}^{-1}\boldsymbol{U}_{\rm unc}(k) - \boldsymbol{V}^T\boldsymbol{H}\boldsymbol{M}\boldsymbol{M}^{-1}\boldsymbol{U}(k)||_2^2 \qquad \text{(from (13)–(15))}\\ &= ||\boldsymbol{V}^T\boldsymbol{H}\boldsymbol{U}_{\rm unc}(k) - \boldsymbol{V}^T\boldsymbol{H}\boldsymbol{U}(k)||_2^2 \quad (\det\boldsymbol{M} = \pm 1)\\ &= ||\boldsymbol{H}\boldsymbol{U}_{\rm unc}(k) - \boldsymbol{H}\boldsymbol{U}(k)||_2^2 \qquad (\boldsymbol{V}^T = \boldsymbol{V}^{-1})\\ &= (\boldsymbol{U}_{\rm unc}(k) - \boldsymbol{U}(k))^T\boldsymbol{H}^T\boldsymbol{H}(\boldsymbol{U}_{\rm unc}(k) - \boldsymbol{U}(k))\\ &= (\boldsymbol{U}_{\rm unc}(k) - \boldsymbol{U}(k))^T\boldsymbol{Q}(\boldsymbol{U}_{\rm unc}(k) - \boldsymbol{U}(k)) \qquad \text{(from (19))}\end{aligned}$$

■

The convex hull in the original space $\mathbf{conv}\,\mathbb{U}$ is trivial since it is a hypercube (n-dimensional cube), centered at the origin and of side length 2. Therefore the projection of $\boldsymbol{U}_{\rm unc}(k)$ on $\mathbf{conv}\,\mathbb{U}$, i.e., $\boldsymbol{U}_{\rm rlx}(k) \in \mathbf{conv}\,\mathbb{U}$ can be found by solving the following *box-constrained* QP

$$\begin{aligned}\underset{\boldsymbol{U}(k)}{\text{minimize}} \quad & (\boldsymbol{U}_{\rm unc}(k) - \boldsymbol{U}(k))^T\boldsymbol{Q}(\boldsymbol{U}_{\rm unc}(k) - \boldsymbol{U}(k))\\ \text{subject to} \quad & \begin{bmatrix}\boldsymbol{I} & \boldsymbol{0}\\ \boldsymbol{0} & -\boldsymbol{I}\end{bmatrix}\begin{bmatrix}\boldsymbol{U}(k)\\ \boldsymbol{U}(k)\end{bmatrix} \preceq \begin{bmatrix}\boldsymbol{1}\\ -\boldsymbol{1}\end{bmatrix},\end{aligned} \tag{16}$$

where $\boldsymbol{I}$ and $\boldsymbol{0}$ are the n-dimensional identity and zero matrices, respectively, $\boldsymbol{1}$ is an n-dimensional vector with all components one, and $\preceq$ denotes the componentwise inequality. Because $\mathbf{conv}\,\mathbb{U}$ is nonempty, problem (16) is by definition

Algorithm 1 Long-Horizon Direct MPC

1: **function** $\boldsymbol{U}^* = \text{DMPCC}(\boldsymbol{U}_{\text{unc}}, \boldsymbol{U}_{\text{ed}}, \boldsymbol{U}_{\text{bab}})$
2: **if** $\boldsymbol{U}_{\text{unc}} \notin \mathbf{conv}\,\mathbb{U}$ **then**
3: $\boldsymbol{U}_{\text{rlx}} \leftarrow \arg\min_{\boldsymbol{U} \in \mathbf{conv}\,\mathbb{U}} ||\boldsymbol{U}_{\text{unc}} - \boldsymbol{U}||_{\boldsymbol{Q}}$
4: $\boldsymbol{U}_{\text{bab}} \leftarrow \lfloor \boldsymbol{U}_{\text{rlx}} \rceil$
5: $\boldsymbol{U}_{\text{unc}} \leftarrow \boldsymbol{U}_{\text{rlx}}$
6: **end if**
7: $\widetilde{\boldsymbol{U}}_{\text{unc}} \leftarrow \widetilde{\boldsymbol{H}}\ \boldsymbol{M}^{-1}\boldsymbol{U}_{\text{unc}}$
8: $\widetilde{\boldsymbol{U}}_{\text{bab}} \leftarrow \boldsymbol{M}^{-1}\boldsymbol{U}_{\text{bab}}$
9: $\widetilde{\boldsymbol{U}}_{\text{ed}} \leftarrow \boldsymbol{M}^{-1}\boldsymbol{U}_{\text{ed}}$.
10: $\rho \leftarrow \min\{\rho_a, \rho_b\}$ ▷ see (9)
11: $\text{SphDec}([\,], \widetilde{\boldsymbol{U}}_{\text{unc}}, 0, n, \rho^2)$
12: $\boldsymbol{U}^* \leftarrow \boldsymbol{M}\ \widetilde{\boldsymbol{U}}^*$
13: **end function**
14:
15: **function** $\widetilde{\boldsymbol{U}}^* = \text{SphDec}(\widetilde{\boldsymbol{U}}, \widetilde{\boldsymbol{U}}_{\text{unc}}, d^2, i, \rho^2)$
16: **for** each $\tilde{u} \in \mathcal{U}$ **do**
17: $\widetilde{U}_i \leftarrow \tilde{u}$
18: $d'^2 \leftarrow ||\ \widetilde{U}_{\text{unc}_i} - \widetilde{\boldsymbol{H}}_{(i,i:n)}\widetilde{\boldsymbol{U}}_{i:n}||_2^2 + d^2$
19: **if** $d'^2 \leq \rho^2$ **then**
20: **if** $i > 1$ **then**
21: $\text{SphDec}(\widetilde{\boldsymbol{U}}, \widetilde{\boldsymbol{U}}_{\text{unc}}, d'^2, i-1, \rho^2)$
22: **else**
23: $\widetilde{\boldsymbol{U}}^* \leftarrow \widetilde{\boldsymbol{U}}$
24: $\rho^2 \leftarrow d'^2$
25: **end if**
26: **end if**
27: **end for**
28: **end function**

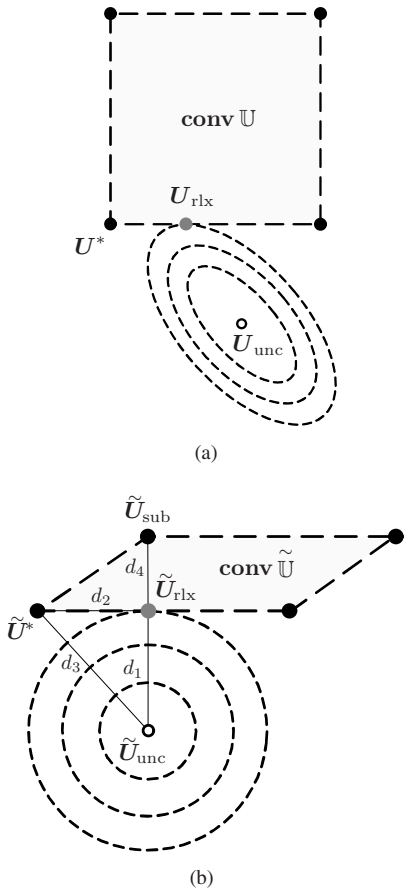


Fig. 3: Convex hull (shown shaded) of four points (shown as black solid circles) in the two-dimensional space. (a) Original space: The level sets of the objective function of problem (16) are shown as dashed ellipses. The projection of $\boldsymbol{U}_{\text{unc}}$ (shown as circle) on $\mathbf{conv}\,\mathbb{U}$ is the point $\boldsymbol{U}_{\text{rlx}}$ (shown as gray solid circle), i.e., the point where the level sets "touch" the convex hull. (b) $\widetilde{\boldsymbol{H}}$-generated space: The level sets of the function of problem (12) are shown as dashed circles. The projection of $\widetilde{\boldsymbol{U}}_{\text{unc}}$ on $\mathbf{conv}\,\widetilde{\mathbb{U}}$ is the point $\widetilde{\boldsymbol{U}}_{\text{rlx}}$. Consider the following case with lengths $d_1 = 0.4$, $d_2 = 0.3$ (thus $d_3 = 0.5$), and $d_4 = 0.2$. Therefore, the lattice point closest to $\widetilde{\boldsymbol{U}}_{\text{unc}}$ (i.e., the optimal solution) is the one labeled as $\widetilde{\boldsymbol{U}}^*$, while the point closest to the projected point $\widetilde{\boldsymbol{U}}_{\text{rlx}}$ is the point labeled as $\widetilde{\boldsymbol{U}}_{\text{sub}}$, which is clearly a suboptimal option.

feasible. Furthermore, it is convex with n optimization variables and $2n$ inequality constraints, thus relatively small and easy to solve [25]. More specifically, a plethora of effective solvers is readily available, and any of iterative projection algorithms, such as [26], [27], gradient projection methods, e.g., [28]–[30], or derivatives [31]–[33] can be employed with negligible differences in computational time, given the small size of the problem at hand.

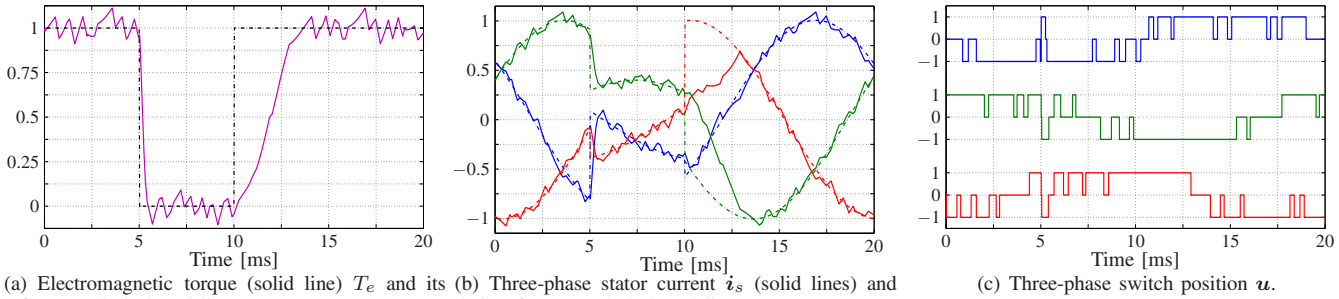
With the projection point $\boldsymbol{U}_{\text{rlx}}$ obtained, the next step is to map it into the $\widetilde{\boldsymbol{H}}$-generated space, i.e., $\widetilde{\boldsymbol{U}}_{\text{rlx}} = \widetilde{\boldsymbol{H}}\ \boldsymbol{M}^{-1}\boldsymbol{U}_{\text{rlx}}$. Subsequently, the radius $\hat{\rho}$ of the new sphere, centered at $\widetilde{\boldsymbol{U}}_{\text{rlx}}$, is computed—as explained above, and the optimization process begins. The pseudocode of the proposed algorithm is presented in Algorithm 1. The initial values of the arguments are computed as explained in Section III.

Projecting the infeasible unconstrained solution on the convex hull (i.e., the feasible set) greatly reduces the computational complexity of the sphere decoder, as shown in Section VI. However, it does not guarantee optimality. This means that the lattice point that is actually closest to $\widetilde{\boldsymbol{U}}_{\text{unc}}(k)$ may not be the one closest to $\widetilde{\boldsymbol{U}}_{\text{rlx}}(k)$. This is explained with an illustrative example in Fig. 3. It is worth mentioning, nonetheless, that thanks to the well-conditioned $\widetilde{\boldsymbol{H}}$ (i.e., the lattice is almost orthogonal and all sides are of comparable lengths) the probability for the optimal solution to be inside the sphere centered at $\widetilde{\boldsymbol{U}}_{\text{rlx}}$ is very high, see Section VI.

V. Computational Complexity

Analyzing the computational complexity of Algorithm 1, the main effort is put into solving (16) (see line 3) and the sphere decoder (lines 15–29). As mentioned in Section IV, problem (16) is a box-constrained QP which means that it can be solved in polynomial time. For example, in [26], [31] it is shown that the proposed schemes exhibit a global convergence rate $O(1/\kappa^2)$, where κ is the number of iterations. Therefore, (16) can be easily solved within 12–18 iterations on average, depending on the size of the problem which varies linearly with the length of the prediction horizon.

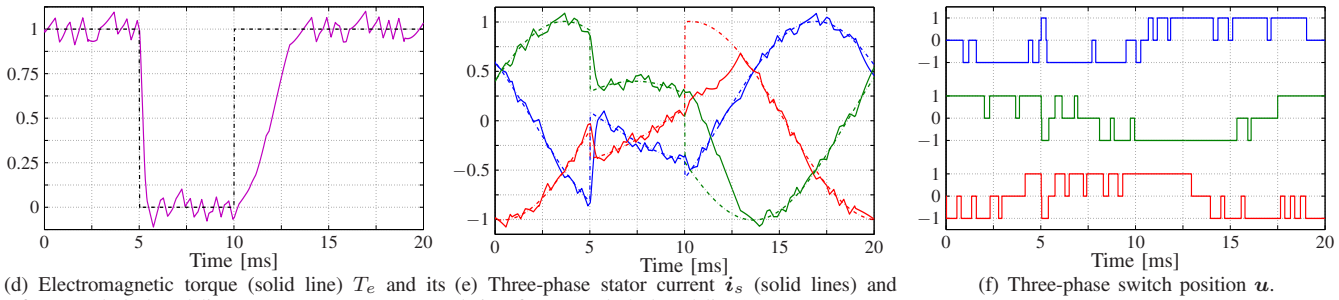
With regards to the computational complexity of the sphere decoding algorithm, this is analyzed based on the flops



(a) Electromagnetic torque (solid line) T_e and its reference (dash-dotted line).

(b) Three-phase stator current $\boldsymbol{i}_s$ (solid lines) and their references (dash-dotted lines).

(c) Three-phase switch position $\boldsymbol{u}$.



(d) Electromagnetic torque (solid line) T_e and its reference (dash-dotted line).

(e) Three-phase stator current $\boldsymbol{i}_s$ (solid lines) and their references (dash-dotted lines).

(f) Three-phase switch position $\boldsymbol{u}$.

Fig. 4: Torque reference steps for direct MPC with a ten-step horizon ($N = 10$) at nominal speed. The sampling interval is $T_s = 25\,\mu$s and the switching frequency is approximately 300 Hz ($\lambda_u = 0.1$). (Upper row) Simulated waveforms with the original sphere decoder (see [2]). (Lower row) Simulated waveforms with the modified sphere decoder (i.e., the proposed approach).

performed in real time. As can be seen in Algorithm 1, when an m-dimensional node, with $m = 1, \ldots, n$, is visited, $n - m + 1$ additions[3] are required for the computation and update of the intermediate (squared) radius d ($n - m$ for the computation of the radius at level m, i.e., $\widetilde{\boldsymbol{H}}_{(m,m:n)}\,\widetilde{\boldsymbol{U}}_{m:n}$, and one for the update $||*||_2^2 + d^2$), see line 18. Moreover one subtraction is required. Regarding the multiplications, only one is required per node (for squaring the Euclidean norm $||*||_2^2$) regardless of its dimension; since $\boldsymbol{U} \in \mathbb{U}$ the result of each one of the $n - m$ multiplications required to compute the product $\widetilde{\boldsymbol{H}}_{(m,m:n)}\widetilde{\boldsymbol{U}}_{m:n}$ is either $\tilde{h}_{m,i}$, $-\tilde{h}_{m,i}$, or 0, with $m \leq i \leq n$. Finally, since the cardinality of the input set $\mathcal{U}$ is three, this implies that for each node visited both its sibling nodes need to be checked to ascertain whether they are inside the hypersphere or not. A more detailed analysis of the operations performed by the direct long-horizon MPC scheme and the employed optimizer can be found in [2], [34].

VI. Performance Evaluation

To obtain the simulation results presented in this section, an MV drive (Fig. 1) consisting of a squirrel cage IM with 3.3 kV rated voltage, 356 A rated current, 2 MVA rated power, 50 Hz nominal frequency, 0.25 p.u. total leakage inductance, and a three-level NPC with constant dc-link voltage $V_{\rm dc} = 5.2$ kV and a fixed neutral point N, is considered. For all cases examined, the controller was operated with the sampling interval $T_s = 25\,\mu$s. All results are shown in the p.u. system.

The performance of the proposed direct MPC scheme is examined during torque transients to examine its dynamical behavior in terms of settling time and reference regulation. The horizon $N = 10$ case is investigated; the weighting factor $\lambda_u = 0.1$ is chosen, such that a switching frequency of approximately 300 Hz results. While operating at rated speed, reference torque steps of magnitude one are imposed. The response of the drive controlled with the sphere decoder in [2] (i.e., optimality is guaranteed) is shown in the upper row of Fig. 4, whereas that of the proposed approach in the lower row of the same figure. As can be seen, the electromagnetic torque in both cases tracks the new desired values as quickly as possible, with the settling time being limited only by the available dc-link voltage (see Figs. 4(a) and 4(d), respectively). In effect the controller behaves like a deadbeat controller. As for the currents, they accurately track their new references (the torque steps on the torque reference are translated into the corresponding current steady-state references), as shown in Figs. 4(b) and 4(e). Finally, in Figs. 4(c) and 4(f) the three-phase switching sequences are shown.

[3]Except when $m = n$, where no additions are performed.

TABLE I: The percentage of times the solution computed by the proposed approach $\boldsymbol{U}_{\rm appl}$ is the (global) optimal solution $\boldsymbol{U}^*$ for different prediction horizons.

Prediction horizon N	$\boldsymbol{U}_{\rm appl} = \boldsymbol{U}^*$ %
1	100
2	100
3	100
4	100
5	99.8
7	99.3
10	98.5

TABLE II: Maximum number of nodes μ explored by (a) the exhaustive search algorithm, (b) the sphere decoder in [2], and (c) the proposed algorithm, during the step-down (μ_d) and step-up (μ_u) torque reference changes shown in Fig. 4 for different prediction horizons.

		Prediction Horizon N						
		1	2	3	4	5	7	10
Exhaustive enumeration	$\max(\mu_d)$	39	1,092	29,523	797,160	21,523,359	$> 1.5 \cdot 10^{10}$	$> 3 \cdot 10^{14}$
Sphere decoder [2]	$\max(\mu_d)$	7	23	43	165	460	1,433	1,760
Proposed approach	$\max(\mu_d)$	5	14	18	26	32	58	92
Exhaustive enumeration	$\max(\mu_u)$	39	1,092	29,523	797,160	21,523,359	$> 1.5 \cdot 10^{10}$	$> 3 \cdot 10^{14}$
Sphere decoder [2]	$\max(\mu_u)$	4	14	36	82	202	1,579	36,092
Proposed approach	$\max(\mu_u)$	3	9	14	18	24	61	114

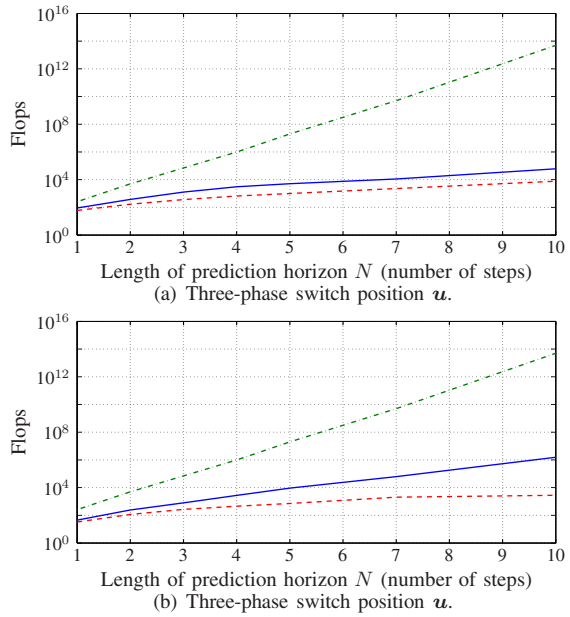


Fig. 5: Maximum flops preformed during the search process when the (a) step-down and (b) step-up torque reference changes in Fig. 4 occur as a function of the prediction horizon N. The solid (blue) line refers to the sphere decoder in [2], the dashed (red) line to the presented algorithm, and the dash-dotted (green) line to the exhaustive enumeration.

In Table I the proposed approach is compared with the sphere decoder in [2] in terms of (sub)optimality for different lengths of the prediction horizon. As can be seen, thanks to the well-conditioned $\widetilde{\boldsymbol{H}}$ optimality is sacrificed only for longer prediction horizons ($N \geq 5$), and even then marginally, without any significant effect on the drive performance, as shown in Fig. 4. As for the computational complexity, a first rough analysis for the two aforementioned techniques and the strategy of exhaustive enumeration in terms of the nodes evaluated as a function of the prediction horizon length is presented in Table II. Moreover, the flops required by these three strategies—when the same conditions are considered—are depicted in Fig. 5. As can be seen, the sphere decoding algorithm with the refinements proposed here significantly reduces the number of examined nodes and real-time flops, even compared with the approach proposed in [2]. For example, the discussed method can visit less nodes and perform less operations by up to 99.7 % when a ten-step horizon and a step-up torque reference change are examined.

VII. Conclusions

This paper proposes refinements for the sphere decoding algorithm employed to solve the long-horizon direct model predictive control (MPC) problem for transient operation. By exploiting the geometry of the underlying quadratic program (QP) a new, tighter sphere is computed that, although it sacrifices optimality (to some degree) when longer horizons are of concern, it can significantly reduce the computational complexity of the integer problem. Thanks to the proposed modifications, the computational burden can be reduced by up to 99.7% for long horizons and a three-level converter, compared to that required for the search algorithm in [2].

Appendix A
Continuous-Time Model of the Drive

The matrices $\boldsymbol{D}$, $\boldsymbol{E}$, and $\boldsymbol{F}$ of the continuous-time state-space model of the drive (3) are

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

$$\boldsymbol{E} = \frac{X_r}{\Phi} \frac{V_{\mathrm{dc}}}{2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad \boldsymbol{F} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

Appendix B
Derivation of the ILS Problem

By introducing $\boldsymbol{Y}(k) = [\boldsymbol{y}^T(k+1) \ \ldots \ \boldsymbol{y}^T(k+N)]^T$ and $\boldsymbol{Y}_{\mathrm{ref}}(k) = [\boldsymbol{y}_{\mathrm{ref}}^T(k+1) \ \ldots \ \boldsymbol{y}_{\mathrm{ref}}^T(k+N)]^T$ to denote the output and the corresponding output reference sequences over the horizon, respectively, function (5) can be written in vector form as

$$J = ||\boldsymbol{\Gamma} \boldsymbol{x}(k) + \boldsymbol{\Upsilon} \boldsymbol{U}(k) - \boldsymbol{Y}_{\mathrm{ref}}||_2^2 + \lambda_u ||\boldsymbol{S} \boldsymbol{U}(k) - \boldsymbol{\Xi} \boldsymbol{u}(k-1)||_2^2, \tag{17}$$

where it was used the fact that $\boldsymbol{Y}(k) = \boldsymbol{\Gamma} \boldsymbol{x}(k) + \boldsymbol{\Upsilon} \boldsymbol{U}(k)$, with the matrices $\boldsymbol{\Upsilon}$, $\boldsymbol{\Gamma}$, $\boldsymbol{S}$ and $\boldsymbol{\Xi}$ being

$$\boldsymbol{\Upsilon} = \begin{bmatrix} \boldsymbol{CBK} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{CABK} & \boldsymbol{CBK} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & & \vdots \\ \boldsymbol{CA}^{N-1}\boldsymbol{BK} & \boldsymbol{CA}^{N-2}\boldsymbol{BK} & \cdots & \boldsymbol{CBK} \end{bmatrix},$$

$$\boldsymbol{\Gamma} = \begin{bmatrix} \boldsymbol{CA} \\ \boldsymbol{CA}^2 \\ \vdots \\ \boldsymbol{CA}^N \end{bmatrix}, \ \boldsymbol{S} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ -\boldsymbol{I} & \boldsymbol{I} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & -\boldsymbol{I} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{I} \end{bmatrix}, \ \boldsymbol{\Xi} = \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{0} \\ \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \end{bmatrix},$$

where $\boldsymbol{0}$ is the zero matrix of appropriate dimensions.

After some algebraic manipulations, (17) takes the form

$$J(k) = ||\boldsymbol{U}(k) + \boldsymbol{Q}^{-1}\boldsymbol{\Lambda}(k)||^2_{\boldsymbol{Q}} + \underbrace{\zeta(k) - \boldsymbol{\Lambda}^T(k)\boldsymbol{Q}^{-T}\boldsymbol{\Lambda}(k)}_{\text{const}(k)} . \tag{18}$$

where

$$\begin{aligned} \zeta(k) &= ||\boldsymbol{\Gamma}\boldsymbol{x}(k) - \boldsymbol{Y}_{\text{ref}}(k)|| + \lambda_u||\boldsymbol{\Xi}\boldsymbol{u}(k-1)|| \,, \\ \boldsymbol{Q} &= \boldsymbol{\Upsilon}^T\boldsymbol{\Upsilon} + \lambda_u\boldsymbol{S}^T\boldsymbol{S} \,, \\ \boldsymbol{\Lambda}(k) &= \boldsymbol{\Upsilon}^T\big(\boldsymbol{\Gamma}\boldsymbol{x}(k) - \boldsymbol{Y}_{\text{ref}}(k)\big) - \lambda_u\boldsymbol{S}^T\boldsymbol{\Xi}\boldsymbol{u}(k-1) \,. \end{aligned}$$

Following, by noticing that $\lambda_u > 0 \Rightarrow \boldsymbol{Q} \succ 0$, then $\boldsymbol{Q}$ can be decomposed as

$$\boldsymbol{Q} = \boldsymbol{H}^T\boldsymbol{H} \,. \tag{19}$$

Using (19) and the unconstrained solution of (18), i.e., $\boldsymbol{U}_{\text{unc}}(k) = -\boldsymbol{Q}^{-1}\boldsymbol{\Lambda}(k)$, and by neglecting the constant term since it is independent of $\boldsymbol{U}(k)$, (18) is written as

$$J = ||\bar{\boldsymbol{U}}_{\text{unc}}(k) - \boldsymbol{H}\boldsymbol{U}(k)||^2_2 \,, \tag{20}$$

which is the ILS function of problem (7). ■

REFERENCES

[1] T. Geyer and D. E. Quevedo, "Multistep finite control set model predictive control for power electronics," *IEEE Trans. Power Electron.*, vol. 29, no. 12, pp. 6836–6846, Dec. 2014.

[2] P. Karamanakos, T. Geyer, and R. Kennel, "Reformulation of the long-horizon direct model predictive control problem to reduce the computational effort," in *Proc. IEEE Energy Convers. Congr. Expo.*, Pittsburgh, PA, Sep. 2014, pp. 3512–3519.

[3] J. B. Rawlings and D. Q. Mayne, *Model Predictive Control: Theory and Design*. Madison, WI: Nob Hill, 2009.

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

[5] T. Geyer, *Model predictive control of high power converters and industrial drives*. Hoboken, NJ: Wiley, 2016.

[6] ——, "Computationally efficient model predictive direct torque control," *IEEE Trans. Power Electron.*, vol. 26, no. 10, pp. 2804–2816, Oct. 2011.

[7] T. Geyer and D. E. Quevedo, "Performance of multistep finite control set model predictive control for power electronics," *IEEE Trans. Power Electron.*, vol. 30, no. 3, pp. 1633–1644, Mar. 2015.

[8] T. Geyer, P. Karamanakos, and R. Kennel, "On the benefit of long-horizon direct model predictive control for drives with *LC* filters," in *Proc. IEEE Energy Convers. Congr. Expo.*, Pittsburgh, PA, Sep. 2014, pp. 3520–3527.

[9] P. Karamanakos, T. Geyer, and R. Kennel, "A computationally efficient model predictive control strategy for linear systems with integer inputs," *IEEE Trans. Control Syst. Technol.*, vol. 24, no. 4, pp. 1463–1471, Jul. 2016.

[10] P. Karamanakos, T. Geyer, N. Oikonomou, F. Kieferndorf, and S. Manias, "Model predictive control in power electronics: Strategies to reduce the computational complexity," in *Proc. IEEE Ind. Electron. Conf.*, Vienna, Austria, Nov. 2013, pp. 5818–5823.

[11] R. Cagienard, P. Grieder, E. C. Kerrigan, and M. Morari, "Move blocking strategies in receding horizon control," *J. of Process Control*, vol. 17, no. 6, pp. 563–570, Jul. 2007.

[12] P. Karamanakos, T. Geyer, and S. Manias, "Direct voltage control of dc-dc boost converters using enumeration-based model predictive control," *IEEE Trans. Power Electron.*, vol. 29, no. 2, pp. 968–978, Feb. 2014.

[13] T. Geyer, G. Papafotiou, and M. Morari, "Model predictive direct torque control—Part I: Concept, algorithm and analysis," *IEEE Trans. Ind. Electron.*, vol. 56, no. 6, pp. 1894–1905, Jun. 2009.

[14] E. L. Lawler and D. E. Wood, "Branch-and-bound methods: A survey," *Op. Res.*, vol. 14, no. 4, pp. 699–719, Jul./Aug. 1966.

[15] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis," *Math. Comput.*, vol. 44, no. 170, pp. 463–471, Apr. 1985.

[16] B. Hassibi and H. Vikalo, "On the sphere-decoding algorithm I. Expected complexity," *IEEE Trans. Signal Proc.*, vol. 53, no. 8, pp. 2806–2818, Aug. 2005.

[17] P. Karamanakos, T. Geyer, and R. Kennel, "Constrained long-horizon direct model predictive control for power electronics," in *Proc. IEEE Energy Convers. Congr. Expo.*, Milwaukee, WI, Sep. 2016, pp. 1–8.

[18] R. Baidya, R. P. Aguilera, P. Acuna, R. Delgado, T. Geyer, D. Quevedo, and T. Mouton, "Fast multistep finite control set model predictive control for transient operation of power converters," in *Proc. IEEE Ind. Electron. Conf.*, Florence, Italy, Oct. 2016, pp. 5039–5045.

[19] S. N. Manias, *Power electronics and motor drive systems*. Cambridge, MA: Academic Press, 2016.

[20] J. Holtz, "The representation of ac machine dynamics by complex signal flow graphs," *IEEE Trans. Ind. Electron.*, vol. 42, no. 3, pp. 263–271, Jun. 1995.

[21] D. Micciancio, "The hardness of the closest vector problem with preprocessing," *IEEE Trans. Inf. Theory*, vol. 47, no. 3, pp. 1212–1215, Mar. 2001.

[22] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász, "Factoring polynomials with rational coefficients," *Math. Ann.*, vol. 261, no. 4, pp. 515–534, 1982.

[23] P. Karamanakos, T. Geyer, and R. Kennel, "Suboptimal search strategies with bounded computational complexity to solve long-horizon direct model predictive control problems," in *Proc. IEEE Energy Convers. Congr. Expo.*, Montreal, QC, Canada, Sep. 2015, pp. 334–341.

[24] L. Babai, "On Lovász' lattice reduction and the nearest lattice point problem," *Combinatorica*, vol. 6, no. 1, pp. 1–13, 1986.

[25] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, UK: Cambridge Univ. Press, 2004.

[26] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $O(1/k^2)$," *Soviet Mathematics Doklady*, vol. 27, no. 2, pp. 372–376, Feb. 1983.

[27] J. Barzilai and J. M. Borwein, "Two-point step size gradient methods," *IMA J. of Numer. Anal.*, vol. 8, no. 1, pp. 141–148, Jan. 1988.

[28] M. Raydan, "On the barzilai and borwein choice of steplength for the gradient method," *IMA J. of Numer. Anal.*, vol. 13, no. 3, pp. 321–326, Jul. 1993.

[29] A. Friedlander, J. M. Martinez, and M. Raydan, "A new method for large-scale box constrained convex quadratic minimization problems," *Optim. Methods and Software*, vol. 5, no. 1, pp. 57–74, Jan. 1995.

[30] A. R. Conn, N. I. M. Gould, and P. L. Toint, "Global convergence of a class of trust region algorithms for optimization with simple bounds," *SIAM J. on Numer. Anal.*, vol. 25, no. 2, pp. 433–460, Mar./Apr. 1988.

[31] A. Beck and M. Teboulle, "Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems," *IEEE Trans. Image Proc.*, vol. 18, no. 11, pp. 2419–2434, Nov. 2009.

[32] B. Hauska, H. J. Ferreau, and M. Diehl, "An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range," *Automatica*, vol. 47, no. 10, pp. 2279–2285, Oct. 2011.

[33] P. Zometa, M. Kögel, T. Faulwasser, and R. Findeisen, "Implementation aspects of model predictive control for embedded systems," in *Proc. Am. Control Conf.*, Montreal, QC, Canada, Jun. 2012, pp. 1205–1210.

[34] P. Karamanakos, T. Geyer, T. Mouton, and R. Kennel, "Computationally efficient sphere decoding for long-horizon direct model predictive control," in *Proc. IEEE Energy Convers. Congr. Expo.*, Milwaukee, WI, Sep. 2016, pp. 1–8.