Suboptimal Search Strategies With Bounded Computational Complexity to Solve Long-Horizon Direct Model Predictive Control Problems

Petros Karamanakos, *Member, IEEE*, Tobias Geyer, *Senior Member, IEEE*, and Ralph Kennel, *Senior Member, IEEE*

***Abstract*—Search algorithms that reduce the time to solve the direct model predictive control (MPC) problem are proposed in this paper. By allowing for suboptimal solutions, the computational complexity of the underlying optimization problem can be significantly reduced, albeit by sacrificing (to a certain degree) optimality. Two approaches are presented and discussed. The first approach requires quadratic time, making it a very efficient candidate for solving the examined problem. Thanks to the second approach, a preset upper limit on the operations performed in real time is not exceeded, thus guaranteeing real-time termination in all runs. To highlight the effectiveness of the introduced strategies, a variable speed drive system with a three-level voltage source inverter is used as an illustrative example.**

I. Introduction

In recent years, acceptance of model predictive control (MPC) [1], [2] by the power electronic community has grown rapidly [3]. The ability of MPC to handle systems with complex dynamics, and/or multiple inputs and outputs—the so-called MIMO systems—and to operate them close to their physical limits, thanks to the underlying constrained optimization problem, enhanced its widespread application. Moreover, the use of a receding horizon introduces feedback to the system, and thus adds a high degree of robustness to the algorithm in the presence of disturbances.

In power electronics, MPC is often implemented as a direct controller to tackle both the output reference tracking and the modulation problems in one stage [3]–[6]. As a result, the switching signals are directly applied to the converter, without first being modulated. However, a pitfall of this approach is that the formulated optimization problem is a (mixed) integer problem, which is NP-hard, i.e., in general very difficult to solve, since its computational complexity increases exponentially with the length of the prediction horizon. A brute-force option to solve this type of problems is to exhaustively enumerate and evaluate all candidate solutions to determine the optimal one. But this enforces, in a way, the use of very short horizons—which are in most cases equal to one step—since for longer horizons the problem becomes computationally intractable. Therefore, computational issues still remain, regardless of the rapid and continued increase in computational power available.

To overcome this issue, several strategies have been proposed that make the real-time implementation of MPC schemes with nontrivial prediction horizons possible [7]. In addition, branch-and-bound techniques [8] can be successfully adopted, as shown in [9]. Lately, the concept of a branch-and-bound algorithm, called sphere decoder [10], [11], has been introduced to the field of power electronics, showing promising results [12]–[14]. By employing one of the aforementioned strategies, one can implement MPC schemes with long prediction horizons, which is a prerequisite to achieving acceptable system performance, especially when high-order systems are concerned [15].

Nevertheless, despite the fact that the above-mentioned algorithms facilitate the implementation of long-horizon MPC schemes, computational challenges still exist since the optimization problem remains NP-hard; its worst-case complexity still grows exponentially. Moreover, the algorithms employed to solve (mixed) integer problems usually do not have real-time guarantees of termination. This can cause many problems in safety critical applications.

In consideration of these pitfalls, the present paper focuses on algorithms that offer shorter solution times, but at the expense of optimality. In particular, suboptimal solutions are occasionally implemented when the operations performed in real time exceed a predefined upper bound—interpreted as the available computational power—while trying to find the optimal solution. In this way, the computational complexity is bounded, or, loosely speaking, fixed, implying that the termination of the algorithm is guaranteed. To further reduce the operations required, but by further sacrificing optimality, the estimated (i.e., guessed) solution of the optimization problem is always implemented, without starting the search process at all. However, as shown in the paper, the estimated solution is in most problem instances the same as the optimal one, thus the performance of the plant does not significantly deteriorate. As a case study, a variable speed drive system is considered, consisting of a three-level neutral point clamped (NPC) voltage source inverter driving a medium-voltage (MV)

P. Karamanakos, and R. Kennel are with the Institute for Electrical Drive Systems and Power Electronics, Technische Universität München, Munich 80333, Germany (e-mail: p.karamanakos@ieee.org, kennel@ieee.org).

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

978-1-4673-7151-3/15/$31.00 ©2015 IEEE

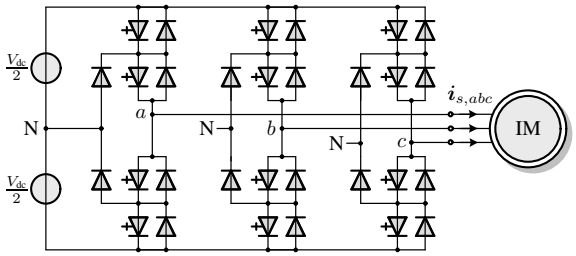


Fig. 1: Three-level three-phase neutral point clamped (NPC) voltage source inverter driving an induction motor (IM) with a fixed neutral point potential.

induction machine (IM), to verify the effectiveness of the proposed approaches.

II. Constrained Optimal Control Problem

In the sequel, the mathematical model of the examined system and the formulation of the optimization problem—and of the equivalent integer least-squares (ILS) problem—are presented.

A. Prediction Model

The first step of formulating and solving an optimization problem underlying MPC problem is to derive an accurate model of the plant to serve as a prediction model. To simplify the calculations it is common to transform the system variables from the three-phase system (abc) to the stationary orthogonal $\alpha\beta$ system, i.e., $\boldsymbol{\xi}_{\alpha\beta} = \boldsymbol{K}\boldsymbol{\xi}_{abc}$, where $\boldsymbol{\xi}_{abc} = [\xi_a \ \xi_b \ \xi_c]^T$ is any variable in the abc plane, and $\boldsymbol{\xi}_{\alpha\beta} = [\xi_\alpha \ \xi_\beta]^T$ the resulting variable in the $\alpha\beta$ plane, via the matrix[1]

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

In Fig. 1 an IM driven by a three-level NPC voltage source inverter with a constant dc-link voltage V_{dc} and a fixed neutral point potential N is depicted. As can be seen, the output voltage of each phase of the inverter can take the three discrete values $-\frac{V_{\text{dc}}}{2}, 0, \frac{V_{\text{dc}}}{2}$, depending on the position of the switches in the respective phase leg, with V_{dc} being the dc-link voltage. These positions are modeled with the integer variables $u_a, u_b, u_c \in \mathcal{U} = \{-1, 0, 1\}$. Introducing $\boldsymbol{u} = [u_a \ u_b \ u_c]^T$ to represent the input vector, and by considering the stator current and the rotor flux in the $\alpha\beta$ plane as the state variables, i.e., $\boldsymbol{x} = [i_{s\alpha} \ i_{s\beta} \ \psi_{r\alpha} \ \psi_{r\beta}]^T$, the continuous-time state-space model of the drive can be written as

$$\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} = \boldsymbol{D}\boldsymbol{x}(t) + \boldsymbol{E}\boldsymbol{u}(t) \tag{1a}$$

$$\boldsymbol{y}(t) = \boldsymbol{F}\boldsymbol{x}(t)\,, \tag{1b}$$

where the output variable is the stator current, i.e., $\boldsymbol{y} = \boldsymbol{i}_{s,\alpha\beta}$. The dynamics, input and output matrices, $\boldsymbol{D} \in \mathbb{R}^{4\times4}$, $\boldsymbol{E} \in \mathbb{R}^{4\times3}$, $\boldsymbol{F} \in \mathbb{R}^{2\times4}$, respectively, are provided in the appendix, along with the detailed derivation of (1). Note that the dynamic of the rotor is neglected since the angular speed ω_r is considered to be a time-varying parameter.

[1]To this end, vectors in the $\alpha\beta$ plane are denoted with the corresponding subscript, unless otherwise stated. If the subscript is omitted, then it is assumed that the vector is in the abc plane.

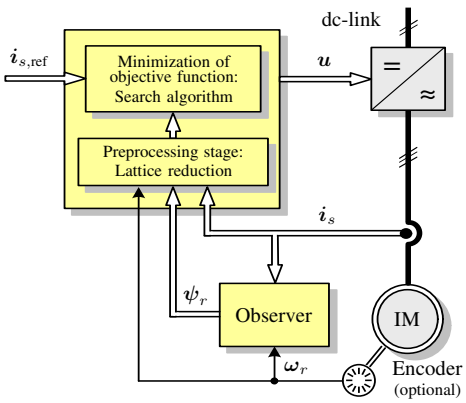


Fig. 2: Model predictive current control with reference tracking for the three-phase three-level NPC inverter with an IM.

Based on the continuous-time model (1), the discrete-time state-space model of the drive can be derived using exact the Euler discretization. This yields

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \tag{2a}$$

$$\boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k)\,, \tag{2b}$$

where the matrices $\boldsymbol{A}, \boldsymbol{B}$ and $\boldsymbol{C}$ are of the form $\boldsymbol{A} = \mathbf{e}^{\boldsymbol{F}T_s}$, $\boldsymbol{B} = -\boldsymbol{F}^{-1}(\boldsymbol{I} - \boldsymbol{A})\boldsymbol{G}$ and $\boldsymbol{C} = \boldsymbol{E}$. Furthermore, $\boldsymbol{I}$ is the identity matrix, $\mathbf{e}$ the matrix exponential, T_s the sampling interval, and $k \in \mathbb{N}$.

B. Direct Model Predictive Control With Current Reference Tracking

The controller discussed in this paper is implemented as a current controller. This means that the main control objective is the accurate tracking of the sinusoidal reference waveform $\boldsymbol{i}_{s,\text{ref},\alpha\beta}$ of the stator current $\boldsymbol{i}_{s,\alpha\beta}$. As shown in Fig. 2, this is achieved without the presence of a modulator, since the switching signals are fed directly to the inverter. As an additional objective, these directly controlled switches are meant to operate at a low switching frequency; this will result in lower switching power losses, which are crucial when MV drives are targeted.

The above-mentioned control tasks are mapped into an objective function of the form

$$J(k) = \sum_{\ell=k}^{k+N-1} ||\boldsymbol{i}_{\text{err},\alpha\beta}(\ell+1|k)||_2^2 + \lambda_u ||\Delta\boldsymbol{u}(\ell|k)||_2^2\,, \tag{3}$$

that penalizes at step k the evolution of the current error $\boldsymbol{i}_{\text{err},\alpha\beta}(\ell+1) = \boldsymbol{i}_{s,\text{ref},\alpha\beta}(\ell+1) - \boldsymbol{i}_{s,\alpha\beta}(\ell+1)$ and the switching (control) effort $\Delta\boldsymbol{u}(\ell) = \boldsymbol{u}(\ell) - \boldsymbol{u}(\ell-1)$ over the finite prediction horizon of length N time-steps. Note that the weighting factor $\lambda_u > 0$ in (3) sets the trade-off between the stator current tracking accuracy (i.e., the deviation of the current from its reference) and the switching effort (i.e., the switching frequency).

In a subsequent step, the objective function (3) is minimized subject to the plant model to determine the sequence of control actions. Thus, the following problem is solved at time-step k

$$\begin{aligned} \underset{\boldsymbol{U}(k)}{\text{minimize}} \quad & J(k) \\ \text{subject to} \quad & \boldsymbol{x}(\ell+1) = \boldsymbol{A}\boldsymbol{x}(\ell) + \boldsymbol{B}\boldsymbol{u}(\ell) \\ & \boldsymbol{y}(\ell) = \boldsymbol{C}\boldsymbol{x}(\ell)\,, \ \forall\, \ell = k, \ldots, k+N-1 \\ & \boldsymbol{U}(k) \in \mathbb{U} \end{aligned} \tag{4}$$

where $\boldsymbol{U}(k) = [\boldsymbol{u}^T(k) \ \boldsymbol{u}^T(k+1) \ \ldots \ \boldsymbol{u}^T(k+N-1)]^T$ is the optimization variable and the feasible set $\mathbb{U}$ is the N-times Cartesian product of the set $\boldsymbol{\mathcal{U}} = \mathcal{U} \times \mathcal{U} \times \mathcal{U} = \mathcal{U}^3$, i.e., $\mathbb{U} = \boldsymbol{\mathcal{U}}^N \subset \mathbb{Z}^n$, with $n = 3N$.

III. Solving the Integer Least-Squares (ILS) Problem

Relaxing the integer constraint $\boldsymbol{U}(k) \in \mathbb{U}$ in (4), i.e., assuming that $\boldsymbol{U}(k) \in \mathbb{R}^n$, then the unconstrained optimal solution $\boldsymbol{U}_{\text{unc}}$ of the problem (4) can be easily found [12]. Based on that unconstrained solution, and after some algebraic manipulations, the optimization problem (4) can be reformulated as [12]

$$\begin{aligned} \underset{\boldsymbol{U}(k)}{\text{minimize}} \quad & ||\bar{\boldsymbol{U}}_{\text{unc}}(k) - \boldsymbol{H}\boldsymbol{U}(k)||_2^2 \\ \text{subject to} \quad & \boldsymbol{U}(k) \in \mathbb{U}\,, \end{aligned} \tag{5}$$

which is an ILS problem due to the integer nature of the decision variable $\boldsymbol{U}$. In (5), $\bar{\boldsymbol{U}}_{\text{unc}}(k) = \boldsymbol{H}\boldsymbol{U}_{\text{unc}}(k) \in \mathbb{R}^n$. Furthermore, the nonsingular, upper triangular matrix $\boldsymbol{H} \in \mathbb{R}^{n\times n}$, is known as the lattice generator matrix and generates the space wherein the integer solution lies, i.e., $\mathcal{L}(\boldsymbol{H}) = \{\sum_{i=1}^{n} \kappa_i \boldsymbol{h}_i \mid \kappa_i \in \mathbb{Z}\}$.

However, the search space as generated by $\boldsymbol{H}$ may need reshaping for solving the optimization problem in a more time-efficient manner. In particular, the lattice should be as orthogonal as possible, whereas the length of the basis vectors of the lattice (i.e., the Euclidean norm of the columns of the lattice generator matrix) should be kept relatively small. To meet these criteria the Lenstra-Lenstra-Lovász (LLL) lattice basis reduction algorithm [16] is employed. The resulting ILS problem takes the form [14]

$$\begin{aligned} \underset{\widetilde{\boldsymbol{U}}(k)}{\text{minimize}} \quad & ||\,\widetilde{\boldsymbol{U}}_{\text{unc}}(k) - \widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{U}}(k)||_2^2 \\ \text{subject to} \quad & \widetilde{\boldsymbol{U}}(k) \in \mathbb{U}\,, \end{aligned} \tag{6}$$

where $\widetilde{\boldsymbol{H}} = \boldsymbol{V}^T\boldsymbol{H}\boldsymbol{M}$ is the reduced lattice generator matrix, as computed by the LLL algorithm, with $\boldsymbol{V} \in \mathbb{R}^{n\times n}$ being an orthogonal matrix and $\boldsymbol{M} \in \{0,1\}^{n\times n}$ a unimodular matrix (i.e., $\det \boldsymbol{M} = \pm 1$), $\widetilde{\boldsymbol{U}}_{\text{unc}}(k) = \widetilde{\boldsymbol{H}}\ \boldsymbol{M}^{-1}\boldsymbol{U}_{\text{unc}}(k)$ and $\widetilde{\boldsymbol{U}}(k) = \boldsymbol{M}^{-1}\boldsymbol{U}(k)$.

A. Optimization Algorithm

To determine the integer solution of problem (6), a variant of the Fincke and Pohst sphere decoding algorithm (see [10]) is employed. As shown in [13] and [14], this search algorithm appears to be computationally very efficient, and thus an excellent candidate for solving the long-horizon MPC problem. This is due to the fact that not all candidate solutions are evaluated, but rather only a very small subset of them, included in a hypersphere (n-dimensional sphere) of radius ρ centered at the unconstrained solution $\widetilde{\boldsymbol{U}}_{\text{unc}}(k)$. Specifically, the sphere decoder is a branch-and-bound search procedure that manages a search tree consisting of m-dimensional nodes, with $m = 1, \ldots, n$; each node represents an element of the integer input vector $\boldsymbol{U}$. This algorithm performs a depth-first search for the optimal solution, meaning that the candidate ILS solutions are found by traversing the n levels of the search tree in a sequential manner until reaching a dead end or the bottom level (i.e., the one-dimensional nodes), where backtracking occurs to examine unvisited nodes at higher levels.

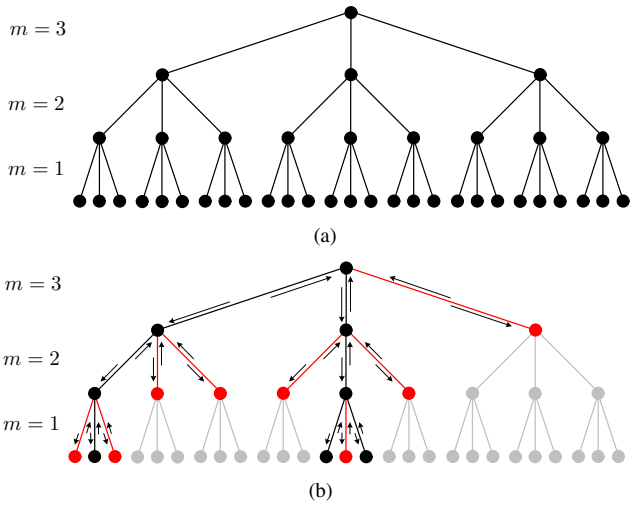


Fig. 3: (a) Search tree of the ILS problem (6) of dimension (depth) $n = 3$. The index $m = \{1, \ldots, n\}$ refers to the level in the search tree. As the search progresses, the dimension of the sphere is reduced from $m = n$ to a one-dimensional sphere. (b) Search tree with nodes explored by the sphere decoder (shown as black solid circles). The nodes that do not lie within the m-dimensional sphere are shown as red solid circles, whereas nodes that are not evaluated at all are depicted as gray solid circles. The direction of the search process is shown with black arrows. To find the optimal solution, nodes are visited with a direction from left to right, and from the higher dimensional layers to the lower ones, until reaching a dead end or the bottom level, where backtracking occurs.

It should be pointed out that in order to reduce the operations required to find the solution, the upper triangular structure of the lattice generator matrix $\widetilde{\boldsymbol{H}}$ is exploited. As a result, the search tree is generated in a bottom-to-top manner, meaning that the top levels of the search tree—which are explored first—consist of the higher-dimensional nodes, whereas the bottom levels visited at the later stages of the search procedure consist of the lower-dimensional nodes (with the one-dimensional nodes being the leaf nodes). In Fig. 3 an example of an integer search tree is illustrated along with the visualization of the search procedure.

As can be understood, the choice of the value for the initial radius ρ is crucial; the radius should be small enough so that the corresponding sphere will include as few lattice points (i.e., nodes) as possible, but not too small to avoid having a sphere

Algorithm 1 Sphere Decoder

```
function Ũ* = SphDec(Ũ, d², i, ρ², Ũ_unc)
    for each ũ ∈ 𝒰 do
        Ũ_i ← ũ
        d'² ← || Ũ_unc_i − H̃_(i,i:n) Ũ_i:n ||₂² + d²
        if d'² ≤ ρ² then
            if i > 1 then
                SphDec(Ũ, d'², i − 1, ρ², Ũ_unc)
            else
                Ũ* ← Ũ
                ρ² ← d'²
            end if
        end if
    end for
end function
```

with no points within, and thus resulting in an empty solution set. In [14], the initial radius is computed based on the Babai estimate [17], [18], i.e., the rounded unconstrained solution to the closest integer vector

$$\boldsymbol{U}_{\text{bab}}(k) = \lfloor \boldsymbol{H}^{-1}\bar{\boldsymbol{U}}_{\text{unc}}(k) \rceil = \lfloor \boldsymbol{U}_{\text{unc}}(k) \rceil \,, \tag{7}$$

and gives a fairly good initial value for the radius

$$\rho(k) = || \, \widetilde{\boldsymbol{U}}_{\text{unc}}(k) - \widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{U}}_{\text{bab}}(k) ||_2 \,, \tag{8}$$

where $\widetilde{\boldsymbol{U}}_{\text{bab}}(k) = \boldsymbol{M}^{-1}\boldsymbol{U}_{\text{bab}}(k)$. Finally, it should be mentioned that when the ILS problem is well-conditioned—thanks to the LLL reduced lattice—the success probability of the Babai estimate, i.e., the probability that the Babai estimate $\boldsymbol{U}_{\text{bab}}(k)$ being equal to the optimal solution $\boldsymbol{U}^*(k)$, becomes higher, as shown in [19]. This implies that the radius of the sphere is the smallest possible, and thus the search tree consists of only a few nodes since the pruning of the suboptimal branches is very efficient, as shown in the next section.

The pseudocode of the sphere decoder is presented in Algorithm 1. The initial values of the arguments are $\hat{\boldsymbol{U}} \leftarrow [\,]$, i.e., the empty vector, $d \leftarrow 0$, $i \leftarrow n$, $\rho \leftarrow \rho(k)$—see (8), and $\widetilde{\boldsymbol{U}}_{\text{unc}} \leftarrow \widetilde{\boldsymbol{H}}\,\boldsymbol{M}^{-1}\boldsymbol{U}_{\text{unc}}(k)$.

IV. Bounded Suboptimal Search Algorithms

To provide a first indication of the computational complexity of the sphere decoder the number of nodes μ examined is used as a metric[2]. As shown in [14] for the ten-step horizon ($N = 10$) case, in the worst-case scenario there are $\max(\mu) = 141$ nodes evaluated by the sphere decoder. However, this number is slightly misleading, since the number of floating point operations (flops) performed depends not only on the number of nodes explored, but also on their dimension, i.e., on the level m the node belongs to. For the same scenario ($N = 10$, $\max(\mu) = 141$), it can be seen in Fig. 4(a) that many nodes are of high dimensions ($m > 15$), implying that few flops are required, due to the upper triangular structure of the lattice generator matrix $\widetilde{\boldsymbol{H}}$ and the direction of the search as performed by the sphere decoder (see Fig. 3). Another observation in Fig. 4(a), is that only two one-dimensional nodes are explored, implying that only two *complete* control (*switching*) sequences $\boldsymbol{U}$ are evaluated in the end; one out of these two is the optimal one.

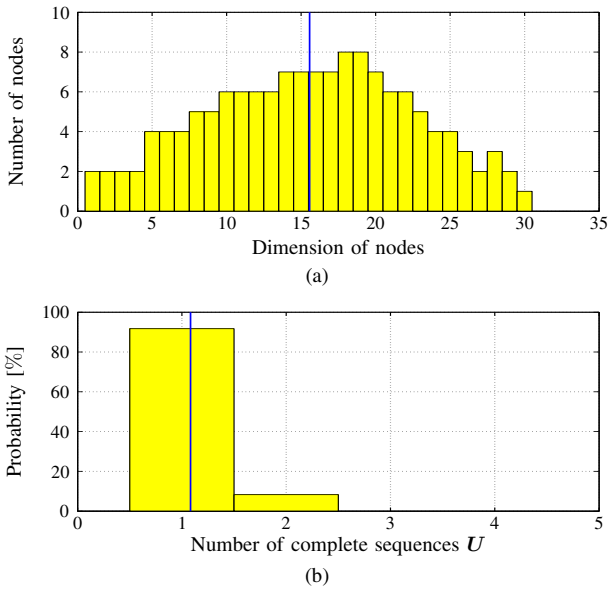


Fig. 4: (a) Dimension of each node visited in the worst-case scenario by the sphere decoder when using a ten-step horizon. The mean dimension of the nodes is indicated by the solid vertical line. (b) Probability distribution of the number of complete control (switching) sequences evaluated by the sphere decoding algorithm, for a ten-step horizon ($N = 10$). The mean number of switching sequences evaluated is equal to 1.083 and it is indicated by the solid vertical line.

Based on the latter observation, Fig. 4(b) depicts the probability distribution of the number of complete control sequences $\boldsymbol{U}$ required to be evaluated at each time-step to decide on the optimal solution for the ten-step horizon case. As can be seen, in only about 8.3% of the cases more than one sequence $\boldsymbol{U}$ is examined. This verifies the argument mentioned in Section III, that when the initial radius ρ is computed based on the Babai estimate, as given by (8), it is as small as possible, or equivalently, the hypersphere that includes the candidate integer solutions is in most cases as tight as possible. Indeed, in about 91.7% of the cases the Babai estimate $\boldsymbol{U}_{\text{bab}}$ is the same as the optimal solution $\boldsymbol{U}^*$, meaning that only the rounded unconstrained solution lies within the hypersphere.

A. Algorithm Based on Estimated Solution

When comparing the radius resulting from the Babai estimate with the radius computed based on an educated guess, as proposed in [12], i.e.,

$$\rho(k) = || \, \widetilde{\boldsymbol{U}}_{\text{unc}}(k) - \widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{U}}_{\text{ed}}(k) ||_2 \,, \tag{9}$$

with $\widetilde{\boldsymbol{U}}_{\text{ed}}(k) = \boldsymbol{M}^{-1}\boldsymbol{U}_{\text{ed}}(k)$ essentially being the shifted previously computed solution $\boldsymbol{U}^*(k-1)$ by one time step and a

[2]The analysis presented hereafter is based on the case study presented in Section V.

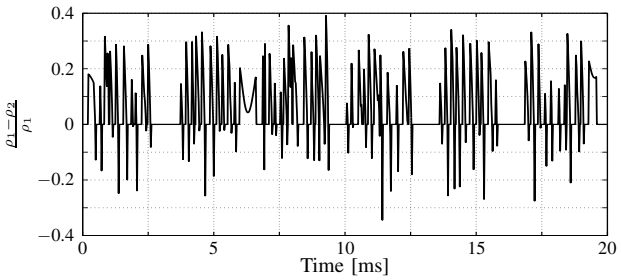


Fig. 5: Relative difference between the initial radii ρ_1 and ρ_2 computed based on the Babai estimate (8) and the educated guess (9), respectively, as a function of time.

repetition of the last switch position, i.e.,

$$\boldsymbol{U}_{\text{ed}}(k) = \begin{bmatrix} \mathbf{0} & \boldsymbol{I} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{I} & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \dots & \dots & \mathbf{0} & \boldsymbol{I} \\ \mathbf{0} & \dots & \dots & \mathbf{0} & \boldsymbol{I} \end{bmatrix} \boldsymbol{U}^*(k-1)\,, \qquad (10)$$

it can be observed that, in a few cases, the latter results in a slightly tighter sphere, see Fig. 5. Therefore, the radius ρ_1 resulting from (8) can be used as the default initial radius of the sphere, but whenever ρ_2 (as computed by (9)) is smaller, then that value could be used instead[3,4]. By doing so, the estimated solution $\boldsymbol{U}_{\text{est}}$, either in the form of $\boldsymbol{U}_{\text{bab}}$, or in the form of $\boldsymbol{U}_{\text{ed}}$ (depending on whether ρ_1 or ρ_2 is used, respectively), is the same as the optimal solution $\boldsymbol{U}^*$ in about 98.7% of the cases. Therefore, the estimated solution $\boldsymbol{U}_{\text{est}}$ could be *always* applied to the plant, with high confidence that in the vast majority of cases it is equal to the optimal solution. As implied, the sphere decoder is not necessary to be used whatsoever, thus the computational complexity of the control algorithm is significantly reduced, but at the expense of a somewhat deteriorated plant performance.

B. Algorithm With Stopping Criterion

As an alternative, in order to keep the performance of the drive system close to the optimal one by applying the optimal solution as often as possible, while keeping the computational burden relatively low, a stopping criterion is added to the sphere decoder. Concretely, an upper bound on the flop count is imposed considering the limited computational power available. This allows one to keep the computational complexity of the algorithm fixed, since the asymptotic running time of the algorithm is bounded by a constant. Whenever the flops performed by the sphere decoder exceed this bound *without* having found a solution, then the last fully computed sequence of control actions $\boldsymbol{U}_{\text{sub}}$ is implemented instead. In the case where a control sequence has not been fully assembled within the given time, then the initial guess is considered as the best integer solution of (6), i.e., $\boldsymbol{U}_{\text{sub}} = \boldsymbol{U}_{\text{est}}$.

This upper bound is set based on the capability of the control hardware. For the sake of the analysis provided hereafter, this upper bound is defined based on the number of operations performed by the exhaustive enumeration algorithm usually used to solve MPC problems in power electronics [3]. Assuming that a two-step MPC algorithm with exhaustive enumeration can be implemented in a microprocessor or a field-programmable gate array (FPGA), then this upper bound is set to $4{,}948$ flops, as shown in Table V in [14]. Therefore, the maximum allowable number of operations performed by the sphere decoder is $4{,}948 - n^2$, since n^2 flops are required for the computation of the unconstrained solution $\widetilde{\boldsymbol{U}}_{\text{unc}}$, based on the analysis given in [14].

The flops performed by the sphere decoder can be summarized as

$$\begin{aligned} N_a &= 3\left(\mu - 1 + \sum_{\nu=1}^{\mu}\big(n - m(\nu)\big)\right), \\ N_s &= 3\mu\,, \\ N_m &= 3\mu\,. \end{aligned} \qquad (11)$$

with N_a, N_s and N_m being the total number of additions, subtractions, and multiplications, respectively. As already mentioned, and as also can be seen in (11), the flops performed depend on both the number of nodes μ examined and their dimension m. In particular, according to Algorithm 1, all the flops are performed for the update of the intermediate radius d' (see Line 4). Hence, for a node at the mth level, $n - m + 1$ additions[5] are required in total, since $n - m$ additions are performed for the computation of $\widetilde{\boldsymbol{H}}_{(m,m:n)}\widetilde{\boldsymbol{U}}_{m:n}$ and one for the addition $||*||_2^2 + d^2$. Moreover, only one multiplication is needed regardless of the dimension of the node; the one for squaring the Euclidean norm $||*||_2^2$. For the computation of the matrix-vector product $\widetilde{\boldsymbol{H}}_{(m,m:n)}\widetilde{\boldsymbol{U}}_{m:n}$ no multiplications are required: since $\widetilde{\boldsymbol{U}} \in \mathbb{U}$ the result of each multiplication $\tilde{h}_{m,i}\tilde{u}_i$, with $m \leq i \leq n$, is one of the values $\{\tilde{h}_{m,i}, -\tilde{h}_{m,i}, 0\}$. Furthermore, only one subtraction is performed per node. Finally, the factor "3" in (11) stems from the fact that for each node explored the two sibling nodes have to be evaluated to determine whether they lie inside of the hypersphere (note that $|\mathcal{U}| = 3$). This is required to get a "certificate" that the solution is found.

The total number of operations $N_t = N_a + N_s + N_m + n^2$ performed in real time, i.e., the operations performed by the sphere decoder plus the n^2 flops required for the computation of the unconstrained solution, is summarized in Table I. Three cases are examined: (a) the optimal solution is always applied, i.e., the sphere decoder is used without any stopping criteria, (b) the estimate of the optimal solution $\boldsymbol{U}_{\text{est}}$ is always applied, i.e., the sphere decoder is inactive and $\boldsymbol{U}_{\text{bab}}$ or $\boldsymbol{U}_{\text{ed}}$ is implemented, and (c) the optimal solution or a suboptimal one is applied, depending on whether the predefined upper

[3]Note that, subscripts "1" and "2" are added in ρ to denote the radius computed by (8) and (9), respectively.

[4]This implies that the initial value of the radius in Algorithm 1 has to be changed from $\rho \leftarrow \rho(k) = \rho_1(k)$, to $\rho \leftarrow \min\{\rho_1(k), \rho_2(k)\}$.

[5]Except when $m = n$, where there are $n - m = 0$ additions.

TABLE I: Average and maximum number of operations N_t performed in real time when applying (a) the optimal solution, (b) the estimated solution, and (c) the optimal or a suboptimal solution, based on the flop count upper bound. The percentage of times the optimal and estimated solutions are the same in each case is also shown.

Prediction	$\boldsymbol{U}^*$		$\boldsymbol{U}_{\text{est}}$			$\boldsymbol{U}^*$ or $\boldsymbol{U}_{\text{sub}}$		
horizon N	avg(N_t)	max(N_t)	avg(N_t)	max(N_t)	$\boldsymbol{U}^* = \boldsymbol{U}_{\text{est}}$ %	avg(N_t)	max(N_t)	$\boldsymbol{U}^* = \boldsymbol{U}_{\text{est}}$ %
1	45	99	9	9	99.4	45	99	99.5
2	139	291	36	36	99.2	139	291	99.4
3	278	501	81	81	98.9	278	501	99.3
4	466	897	144	144	98.5	466	897	99.3
5	702	1,587	225	225	97.9	702	1,587	99.2
7	1,321	3,030	441	441	97.0	1,321	3,030	99.1
10	2,715	8,268	900	900	95.7	2,673	4,849	98.7

bound count is violated or not, i.e., the modified sphere decoder with the stopping criterion is used. Specifically, the table shows the average and maximum number of operations N_t performed in real time for different prediction horizons[6]. The weighting factor λ_u is tuned such that the converter operates at a switching frequency of about 300 Hz in all cases examined, regardless of the prediction horizon. In addition, the percentage of times the optimal solution $\boldsymbol{U}^*$ is equal to the estimated solution $\boldsymbol{U}_{\text{est}}$ is shown. This can be interpreted as the success probability of the initial guess.

As can be seen in Table I, when a ten-step horizon is used and the estimated solution is always applied, then the estimated and optimal solutions are the same in 95.7% of the cases. Thus, a suboptimal solution is essentially implemented in only 4.3% of the cases, implying that the plant performance is only slightly deteriorated, as shown in Section V. On the other hand, and for the same case examined ($N = 10$), when a suboptimal solution is occasionally implemented, then the initial guess is equal to the solution in 98.7% of the cases. However, a suboptimal solution is applied in even less cases, since not in all these 1.3% of the cases the predefined flop count upper bound is violated; the sphere decoder manages to find the optimal solution before hitting the bound. Actually, a suboptimal solution $\boldsymbol{U}_{\text{sub}}$ is applied in 0.9% of the cases, implying that the performance is only marginally suboptimal. This can be seen in Table II, where the percentage of times the optimal solution is applied is shown.

TABLE II: The percentage of times the optimal solution is applied when applying (a) the estimated solution, and (b) either the optimal or a suboptimal solution, based on the flop count upper bound, depending on the length of the prediction horizon N.

Prediction	$\boldsymbol{U}_{\text{est}}$	$\boldsymbol{U}^*$ or $\boldsymbol{U}_{\text{sub}}$
horizon N	$\boldsymbol{U}_{\text{applied}} = \boldsymbol{U}^*$ %	
1	99.4	100
2	99.2	100
3	98.9	100
4	98.5	100
5	97.9	100
7	97.0	100
10	95.7	99.1

V. 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 $V_{\text{dc}} = 5.2\,\text{kV}$ and a fixed neutral point N, is considered. The rated values of the induction machine and the parameters of the drive are summarized in Table III both in SI quantities and in the p.u. system. 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.

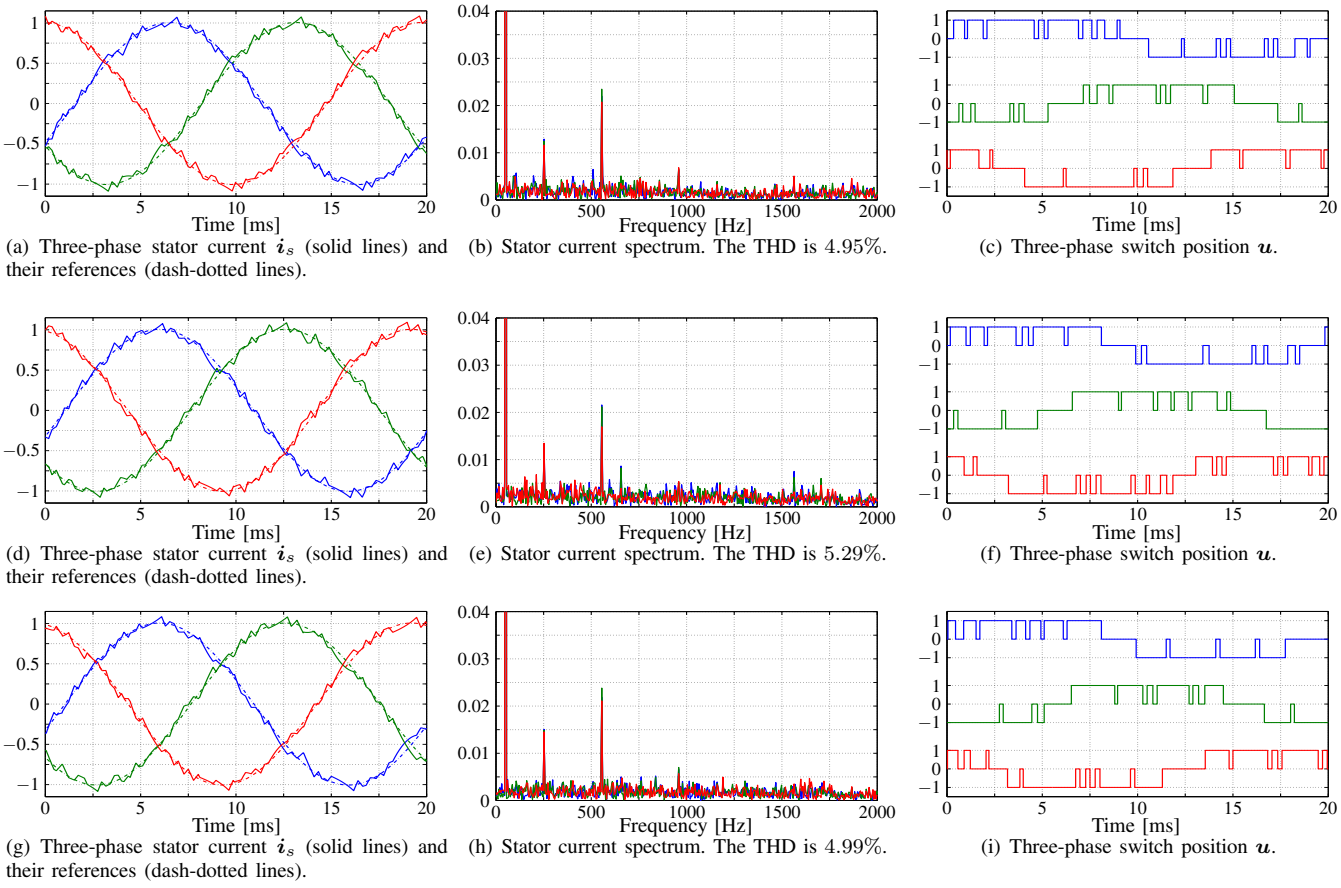
[6]The maximum number of flops $\max(N_t)$ is derived based on the worst-case scenario, i.e., the case where (a) the maximum number of nodes $\max(\mu)$ is explored, and (b) $\boldsymbol{U} \in \mathbb{U} \setminus \{0\}$, since multiplications with zero result in a decrease in the number of additions.

In Fig. 6 the steady-state performance of the drive is depicted when the optimal and the two suboptimal search strategies described are used to solve the MPC problem. In particular, Figs. 6(a)—6(c) show the behavior of the drive when the optimal solution is always applied (i.e., the sphere decoder always finds a solution), for Figs. 6(d)—6(f) the system performance is examined when the estimated solution is implemented (i.e., the sphere decoder is not used), whereas in Figs. 6(g)—6(i) the same scenario is shown when the sphere decoder with the stopping criterion is used. A ten-step horizon ($N = 10$) MPC is tested. The switching frequency of the drive is about 300 Hz, after appropriately tuning the weighting factor λ_u. The three-phase stator current waveforms with their references, and the resulting current spectrum are depicted in Figs. 6(a), 6(d) and 6(g), and Figs. 6(b), 6(e) and 6(h), respectively. As can be seen, when the sphere decoder is not used, the total harmonic distortion (THD) of the current is with 5.29% relatively low, considering the low switching frequency and the fact that the estimated and optimal solutions are equal in 95.7% of the cases. With regards to the sphere decoder with the stopping criterion, the THD is 4.99%, thus very close to the optimal one, which is 4.95%. Finally, the three-phase switching sequences are shown in Figs. 6(c), 6(f) and 6(i).

Table IV summarizes the resulting THD for the three cases discussed in Table I for different lengths of the prediction horizon N. The weighting factor λ_u is tuned such that a switching frequency of about 300 Hz results. As can be seen, when the estimated solution is implemented, the THD is slightly worse compared to that resulting from the optimal case. However, the delivered algorithm is much less com-

TABLE III: Rated values and parameters of the drive

Rated values		Parameters	
Induction Motor			
Voltage	3,300 V	Stator resistance (R_s)	57.61 mΩ (0.0108 p.u.)
Current	356 A	Rotor resistance (R_r)	48.89 mΩ (0.0091 p.u.)
Real power	1.587 MW	Stator leakage reactance (X_{ls})	2.544 mH (0.1493 p.u.)
Apparent power	2.035 MVA	Rotor leakage reactance (X_{lr})	1.881 mH (0.1104 p.u.)
Stator frequency	50 Hz	Mutual reactance (L_m)	40.01 mH (2.349 p.u.)
Rotational speed	596 rpm		
Inverter			
		Dc-link voltage (V_{dc})	5.2 kV (1.930 p.u.)



(a) Three-phase stator current $\boldsymbol{i}_s$ (solid lines) and their references (dash-dotted lines).
(b) Stator current spectrum. The THD is 4.95%.
(c) Three-phase switch position $\boldsymbol{u}$.
(d) Three-phase stator current $\boldsymbol{i}_s$ (solid lines) and their references (dash-dotted lines).
(e) Stator current spectrum. The THD is 5.29%.
(f) Three-phase switch position $\boldsymbol{u}$.
(g) Three-phase stator current $\boldsymbol{i}_s$ (solid lines) and their references (dash-dotted lines).
(h) Stator current spectrum. The THD is 4.99%.
(i) Three-phase switch position $\boldsymbol{u}$.

Fig. 6: Simulated waveforms produced by the optimal and the two suboptimal search algorithms employed to solve the direct model predictive control problem with current reference tracking at steady-state operation, at full speed and rated torque. (a)—(c) The optimal solution is always applied, (d)—(f) the estimated solution is always applied, and (g)—(i) either the optimal or a suboptimal solution is applied, based on the flop count upper bound. A ten-step horizon ($N = 10$) is used, and the sampling interval is $T_s = 25\,\mu$s. The weighting factor λ_u is tuned such that a switching frequency of approximately 300 Hz results in all cases examined.

putationally expensive since its running time is $O(n^2)$, thus longer prediction horizons can be implemented and the system performance can be improved. As an example, consider the suboptimal MPC with a ten-step horizon and the optimal MPC with a four-step horizon. As can be seen in Table I, these two algorithms are equally computationally expensive since 900 flops are required for the former algorithm, whereas $\max(N_t) = 897$ flops for the latter. The resulting THD, though, is 5.29% when the estimated solution is applied, which is less than the 5.37% THD produced by MPC when the optimal solution is implemented (see Table IV). On the other hand, when the sphere decoder with the stopping criterion is implemented, the performance of the drive remains close to the optimal one, but the computations required in the worst-case scenario are reduced by approximately 41%, see the $N = 10$ case in Table I.

VI. Conclusions

This paper proposes suboptimal search algorithms with bounded computational complexity for solving the direct

TABLE IV: Resulting THD when applying (a) the optimal solution, (b) the estimated solution, and (c) the optimal or a suboptimal solution, based on the flop count upper bound, for different lengths of the prediction horizon N.

Prediction Horizon N	$\boldsymbol{U}^*$	$\boldsymbol{U}_{\text{est}}$	$\boldsymbol{U}^*$ or $\boldsymbol{U}_{\text{sub}}$
	THD %		
1	5.76	5.83	5.76
2	5.65	5.76	5.65
3	5.43	5.68	5.43
4	5.37	5.59	5.37
5	5.29	5.54	5.29
7	5.09	5.38	5.09
10	4.95	5.29	4.99

model predictive control (MPC) problem with current reference tracking in a computationally efficient manner. By implementing sometimes suboptimal solutions the complexity of the proposed MPC scheme is kept low, without sacrificing optimality too much. Two approaches are proposed. According to the first one, a search algorithm for finding the optimal solution is not required; the estimated solution, found in quadratic time, is always implemented. With regards to the second approach, a modified sphere decoder with a stopping criterion—namely, an upper bound on the operations performed in real time—is proposed. In this way, the computational complexity is fixed, whereas the plant performance is almost optimal.

APPENDIX

The output voltage of the inverter—which is the same as the stator voltage $\boldsymbol{v}_{s,\alpha\beta}$—is given by

$$\boldsymbol{v}_{\alpha\beta} = \frac{V_{\text{dc}}}{2}\,\boldsymbol{u}_{\alpha\beta} = \frac{V_{\text{dc}}}{2}\boldsymbol{K}\boldsymbol{u}\,. \tag{12}$$

Considering (12) and the drive parameters, i.e., the stator R_s and rotor R_r resistances, and the stator X_{ls}, rotor X_{lr} and mutual X_m reactances, the state equations are[7] [20]

$$\frac{\mathrm{d}\boldsymbol{i}_s}{\mathrm{d}t} = -\frac{1}{\tau_s}\boldsymbol{i}_s + \left(\frac{1}{\tau_r} - \omega_r \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\right)\frac{X_m}{\Phi}\boldsymbol{\psi}_r + \frac{X_r}{\Phi}\boldsymbol{v}_s \tag{13a}$$

$$\frac{\mathrm{d}\boldsymbol{\psi}_r}{\mathrm{d}t} = \frac{X_m}{\tau_r}\boldsymbol{i}_s - \frac{1}{\tau_r}\boldsymbol{\psi}_r + \omega_r \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\boldsymbol{\psi}_r \tag{13b}$$

where $\Phi = X_s X_r - X_m^2$, $X_s = X_{ls} + X_m$, $X_r = X_{lr} + X_m$, and the stator and rotor time constants τ_s and τ_r, are given by $\tau_s = X_r\Phi/(R_s X_r^2 + R_r X_m^2)$ and $\tau_r = X_r/R_r$, respectively.

Thus, the matrices $\boldsymbol{D}$, $\boldsymbol{E}$, and $\boldsymbol{F}$ in (1) are

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

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

[7] In (13) all vectors are in the $\alpha\beta$ plane, thus subscripts are dropped for convenience.

REFERENCES

[1] J. M. Maciejowski, *Predictive Control with Constraints*. Englewood Cliffs, NJ: Prentice-Hall, 2002.

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

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

[4] J. Rodríguez, M. P. Kazmierkowski, J. R. Espinoza, P. Zanchetta, H. Abu-Rub, H. A. Young, and C. A. Rojas, "State of the art of finite control set model predictive control in power electronics," *IEEE Trans. Ind. Informat.*, vol. 9, no. 2, pp. 1003–1016, May 2013.

[5] T. Geyer, "Low complexity model predictive control in power electronics and power systems," Ph.D. dissertation, Autom. Control Lab. ETH Zurich, Zurich, Switzerland, 2005.

[6] P. Karamanakos, "Model predictive control strategies for power electronics converters and ac drives," Ph.D. dissertation, Elect. Mach. and Power Electron. Lab. NTU Athens, Athens, Greece, 2013.

[7] 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 Ind. Electron. Mag.*, vol. 8, no. 1, pp. 32–43, Mar. 2014.

[8] 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.

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

[10] 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.

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

[12] 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.

[13] ——, "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.

[14] 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.

[15] 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.

[16] 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.

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

[18] M. Grotschel, L. Lovász, and A. Schriver, *Geometric Algorithms and Combinatorial Optimization*, 2nd ed. New York: Springer-Verlag, 1993.

[19] X.-W. Chang, J. Wen, and X. Xie, "Effects of the LLL reduction on the success probability of the Babai point and on the complexity of sphere decoding," *IEEE Trans. Inf. Theory*, vol. 59, no. 8, pp. 4915–4926, Aug. 2013.

[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.