Computationally Efficient Sphere Decoding for Long-Horizon Direct Model Predictive Control

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

***Abstract*—In this paper we present a computationally efficient implementation of the sphere decoder, which is employed to solve the integer least-squares (ILS) problem underlying direct model predictive control (MPC) for power electronic applications. The introduced modifications take advantage of the structure of the problem and, as a result, the required real-time operations can be reduced to a minimum. The efficacy of the developed algorithm is demonstrated with a variable speed drive system with a three-level voltage source inverter.**

I. Introduction

Model predictive control (MPC) [1] is a control method emerged in the process industry in the 1970s. MPC solves a constrained optimization problem in an open-loop fashion, and adopts the notion of the receding horizon to provide feedback and robustness. Moreover, MPC is formulated in the time—rather in the frequency—domain, and this enables it to address multiple-input multiple-output (MIMO) systems with complex, nonlinear dynamics.

Over the last decade researchers within the power electronics community have extensively used MPC to effectively tackle the modern, advanced control challenges they deal with [2], [3]. To simplify the controller design, MPC schemes are frequently implemented as direct controllers, meaning that the computation and generation of the switching commands is performed in one computational stage, thus bypassing the subsequent modulation [4]–[6].

Nonetheless, a disadvantage of this approach is that the underlying optimization problem is a (mixed) integer problem, i.e., an NP-hard problem. This means that the computational complexity increases exponentially with the length of the prediction horizon and the number of the decision variables (i.e., the control inputs), and as a result the problem might become computationally intractable. Moreover, exhaustive enumeration of all candidate solutions which is usually performed to solve the direct MPC problems in power electronics further exaggerates the computational cost. To avoid the aforementioned subtleties and pitfalls, it is common practice to implement very short horizons, with most of cases being equal to one step. However, this is not the recommended solution, since long horizons improve the system performance, especially when more complex (e.g., higher-order) systems are of concern [7].

To this end, several strategies have been proposed that allow the real-time implementation of MPC schemes with long, nontrivial horizons [8]. As an alternative, branch-and-bound methods [9] have shown to be reasonably effective when tailored to a specific MPC problem [10]. The sphere decoder [11], [12] is such an algorithm that achieves a significant reduction of the computational burden thanks to its smart bounding principle. Recently, several works highlighted its effectiveness when adopted in linear systems with integer inputs [13], [14], such as power electronics [15]–[18].

To facilitate the implementation of the sphere decoder in a real-time system, such as a field programmable gate array (FPGA), a further reduction in the number of operations performed in real time is desirable. This paper proposes refinements, both in the branching and backtracking procedures of the sphere decoder, that achieve two goals. First, a tighter upper bound on the number of candidate solutions is provided. Second, by exploiting the structure of the underlying optimization problem the real-time operations are significantly reduced. The effectiveness of these modifications is highlighted with the chosen case study, i.e., 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). As will be shown, the computational complexity of the optimization problem can be reduced by more than 55% when long horizons, such as ten steps, are considered.

II. Optimal Control Problem

For comparison purposes with [17] and [15], in this work the optimal control problem is formulated as a current control problem of 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 (Fig. 1).

A. Prediction Model

Before deriving the mathematical model of the plant that serves as the prediction model, the system variables from the three-phase system (abc) are transformed to the stationary orthogonal $\alpha\beta$ system. As a result, any variable in the abc-plane $\boldsymbol{\xi}_{abc} = [\xi_a \ \xi_b \ \xi_c]^T$, can be mapped into a two-dimensional 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

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

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

T. Mouton is with the Department of Electrical and Electronic Engineering, University of Stellenbosch, 7602 Stellenbosch, South Africa; e-mail: dtmouton@sun.ac.za

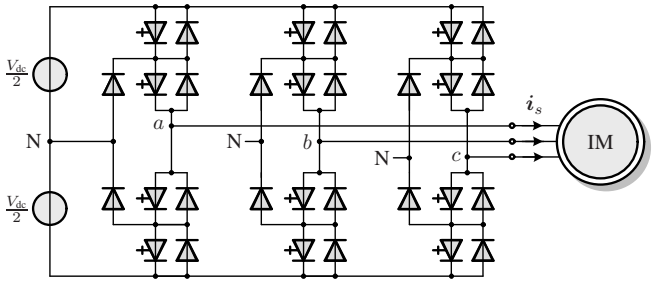


Fig. 1: Three-level three-phase neutral point clamped (NPC) voltage source inverter driving an induction motor (IM) with a fixed neutral point potential.

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

To describe the output voltage of the inverter the integer variable $u_x \in \mathcal{U} = \{-1, 0, 1\}$ is introduced that models the switch position in phase $x \in \{a, b, c\}$. Depending on the value of u_x the inverter can produce three discrete voltage levels, i.e., $-\frac{V_{\mathrm{dc}}}{2}$, 0, $\frac{V_{\mathrm{dc}}}{2}$, at the x-phase terminal. Aggregating the three-phase switch positions in the input vector $\boldsymbol{u} = [u_a\ u_b\ u_c]^T \in \boldsymbol{\mathcal{U}} = \mathcal{U} \times \mathcal{U} \times \mathcal{U} = \mathcal{U}^3$, the inverter output voltage is given by[1]

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

For modeling 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 are chosen as state variables, whereas the dynamic of the rotor angular speed ω_r is neglected, since the speed is considered to be a time-varying parameter. Moreover, the machine input is the stator voltage $\boldsymbol{v}_{s,\alpha\beta}$, which is given by (1), since it is equal to the output inverter voltage. Consequently, the continuous-time state equations are[2] [19]

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

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

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

where R_s (R_r) is the stator (rotor) resistance, and X_{ls} (X_{lr}) and X_m the stator (rotor) and mutual reactance. Based on these model parameters the constants $\Phi = X_s X_r - X_m^2$, $X_s = X_{ls} + X_m$ and $X_r = X_{lr} + X_m$ are introduced, whereas the stator and rotor time constants are defined as $\tau_s = X_r\Phi/(R_s X_r^2 + R_r X_m^2)$ and $\tau_r = X_r/R_r$, respectively. Moreover, H is the inertia, and T_ℓ and T_e denote the mechanical load and electromagnetic torque, respectively. Finally, $\boldsymbol{I}$ is the identity matrix of appropriate dimension (here 2×2).

[1]To this end, vectors in the $\alpha\beta$-plane are denoted with the corresponding subscript. For vectors in the abc-plane the subscript is omitted.

[2]In (2) all vectors are in the $\alpha\beta$-plane, and the subscripts are dropped for convenience.

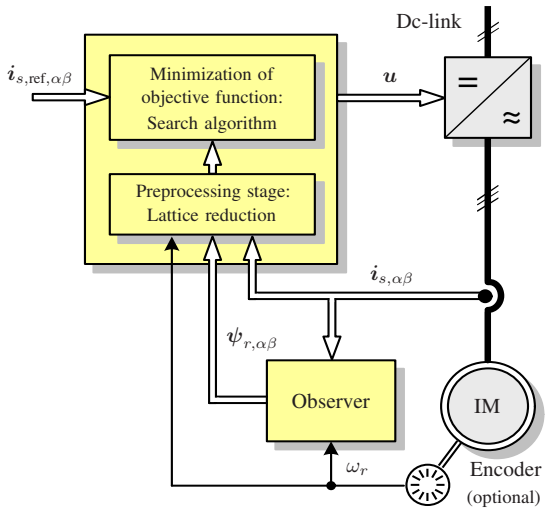


Fig. 2: Model predictive current control with reference tracking for the three-phase three-level NPC inverter with an induction machine.

In a next step, the state equations (2) are arranged as a set of first-order differential equations in order to derive the state-space model in the continuous-time domain. To do so, the state vector $\boldsymbol{x} = [i_{s\alpha}\ i_{s\beta}\ \psi_{r\alpha}\ \psi_{r\beta}]^T$ is introduced, encompassing the stator current and the rotor flux in the $\alpha\beta$-plane; the three-phase switch position $\boldsymbol{u} = [u_a\ u_b\ u_c]^T$ forms the input vector, and the stator current is the output variable, i.e., $\boldsymbol{y} = \boldsymbol{i}_{s,\alpha\beta}$. Thus, the drive can be described by

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

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

where the continuous-time matrices $\boldsymbol{D}$, $\boldsymbol{E}$, and $\boldsymbol{F}$ are given in the appendix.

Discretizing (3) using exact Euler discretization, the discrete-time state-space model of the drive takes the form

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

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

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

B. Direct Model Predictive Control With Current Reference Tracking

The main control objective of the presented control algorithm is for the stator current $\boldsymbol{i}_{s,\alpha\beta}$ to accurately track its reference $\boldsymbol{i}_{s,\mathrm{ref},\alpha\beta}$ by appropriately manipulating the inverter switches. This is to be achieved at low switching frequencies for reduced switching power losses, which, in MV drives, dominate over the conduction losses. To meet the aforementioned objectives, the controller both computes and directly applies the switching signals to the inverter in one stage, thus a subsequent modulation stage is not required, see Fig. 2.

Considering the above-mentioned control objectives two terms are introduced: the current (i.e., output) tracking error term $\boldsymbol{i}_{s,\text{err},\alpha\beta}(k) = \boldsymbol{i}_{s,\text{ref},\alpha\beta}(k) - \boldsymbol{i}_{s,\alpha\beta}(k)$ and the switching (i.e., control) effort term $\Delta\boldsymbol{u}(k) = \boldsymbol{u}(k) - \boldsymbol{u}(k-1)$. Then, the objective function

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

penalizes at step k the evolution of these two terms over the finite prediction horizon of length N time steps. Note that the evolution of the second term is weighted by a factor $\lambda_u > 0$. This factor sets the trade-off between the two terms in (5), i.e., the trade-off between the deviation of the current from its reference and the switching frequency f_{sw}.

Having defined the objective function (5), and based on the plant model (4), the optimal sequence of control actions $\boldsymbol{U}^*(k) = [\boldsymbol{u}^{*T}(k)\ \boldsymbol{u}^{*T}(k+1) \dots \boldsymbol{u}^{*T}(k+N-1)]^T$ is acquired by solving the following problem

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

where $\boldsymbol{U}(k) = [\boldsymbol{u}^T(k)\ \boldsymbol{u}^T(k+1) \dots \boldsymbol{u}^T(k+N-1)]^T$ 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. Formulating and Solving the Integer Least-Squares Problem

As shown in [15], problem (6) can be written as the ILS problem (the optimization variable $\boldsymbol{U}$ is integer) of the form

$$\underset{\boldsymbol{U}(k)\in\mathbb{U}}{\text{minimize}} \quad ||\bar{\boldsymbol{U}}_{\text{unc}}(k) - \boldsymbol{H}\boldsymbol{U}(k)||_2^2 \,. \quad (7)$$

In (7), $\bar{\boldsymbol{U}}_{\text{unc}}(k) = \boldsymbol{H}\boldsymbol{U}_{\text{unc}}(k) \in \mathbb{R}^n$, with $\boldsymbol{U}_{\text{unc}} \in \mathbb{R}^n$ being the unconstrained solution to (6), i.e., the solution acquired after relaxing the feasible set from $\mathbb{U}$ to $\mathbb{R}^n$. Moreover, the nonsingular, upper triangular matrix $\boldsymbol{H} \in \mathbb{R}^{n\times n}$ is referred to as the lattice generator matrix and it generates the discrete space wherein the solution lies.

For an ILS problem to be solved in a time-efficient manner, the lattice should be as close to orthogonal as possible, and the length of the basis vectors that generate it (i.e., the Euclidean norm of the columns of the lattice generator matrix) should be relatively small. Problem (7) is likely to be ill-conditioned since the lattice generated by $\boldsymbol{H}$ might not meet these criteria. Therefore, it is common practice to transform the initial lattice generator matrix (i.e., $\boldsymbol{H}$) to a new matrix that will allow speeding up the optimization stage. By employing the Lenstra-Lenstra-Lovász (LLL) lattice basis reduction algorithm [20] a reduced lattice generator matrix $\widetilde{\boldsymbol{H}}$ is computed, and the equivalent ILS problem becomes [17]

$$\underset{\boldsymbol{U}(k)\in\mathbb{U}}{\text{minimize}} \quad ||\,\widetilde{\boldsymbol{U}}_{\text{unc}}(k) - \widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{U}}(k)||_2^2 \,, \quad (8)$$

with $\widetilde{\boldsymbol{U}}_{\text{unc}}(k) = \widetilde{\boldsymbol{H}}\,\hat{\boldsymbol{U}}_{\text{unc}}(k)$, $\hat{\boldsymbol{U}}_{\text{unc}}(k) = \boldsymbol{M}^{-1}\boldsymbol{U}_{\text{unc}}(k)$, $\widetilde{\boldsymbol{U}}(k) = \boldsymbol{M}^{-1}\boldsymbol{U}(k)$ 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 $\boldsymbol{M} \in \{0,1\}^{n\times n}$ unimodular (i.e., $\det \boldsymbol{M} = \pm 1$).

To efficiently solve the ILS problem (8) underlying the long-horizon MPC problem, the branch-and-bound method called sphere decoding [11] is employed. According to the principle of this algorithm, the search for the optimal solution is limited to a hypersphere (n-dimensional sphere) of radius ρ centered at the unconstrained solution $\widetilde{\boldsymbol{U}}_{\text{unc}}(k)$. Hence, not all candidate solutions (i.e., n-dimensional lattice points) have to be evaluated, but only those that are included in the computed hypersphere. As shown in [16] and [17], the candidate solutions that are eventually evaluated form a very small subset of all possible solutions, meaning that the proposed optimizer significantly outperforms the brute-force approach of the exhaustive enumeration in terms of required computational resources.

As implied above, the upper bound in the branch-and-bound mechanism of sphere decoder is defined by the value for the radius ρ. Therefore, it is very important to compute as small an initial radius ρ as possible in order to have a tight—but nonempty—sphere that will include the smallest possible (nonzero) number of lattice points (i.e., nodes). As proposed in [21], the initial radius is computed based on an estimated solution $\widetilde{\boldsymbol{U}}_{\text{est}} = \boldsymbol{M}^{-1}\boldsymbol{U}_{\text{est}}$, and it is given by

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

with

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

and

$$\rho_b(k) = ||\,\widetilde{\boldsymbol{U}}_{\text{unc}}(k) - \widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{U}}_{\text{ed}}(k)||_2\,. \quad (11)$$

In (10), the estimated solution is the so-called Babai estimate [22], [23], i.e., $\boldsymbol{U}_{\text{est}} = \boldsymbol{U}_{\text{bab}}$, which is the rounded unconstrained solution to the closest integer vector,

$$\boldsymbol{U}_{\text{bab}}(k) = \lfloor \boldsymbol{H}^{-1}\bar{\boldsymbol{U}}_{\text{unc}}(k) \rceil = \lfloor \boldsymbol{U}_{\text{unc}}(k) \rceil\,. \quad (12)$$

On the other hand, in (11), $\boldsymbol{U}_{\text{est}} = \boldsymbol{U}_{\text{ed}}$, which is 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 [15].

Having computed the initial radius based on (9), the search for the optimal solution begins. This is done by traversing the n levels of the generated search tree—consisting of m-dimensional nodes, $m = 1, \dots, n$, and branches that constitute the candidate elements of the solution—in a depth-first search manner. The goal is to reach the bottom level of the tree[3] (i.e., the leaf nodes) as fast as possible so that to update the candidate solution; at that point, the radius of the hypersphere is also updated resulting in a tighter sphere. What should also

[3] As in [17], in this work the top levels of the search tree—which are explored first—consist of the higher-dimensional lattice points, whereas the bottom levels, visited at the later stages of the search procedure, consist of the lower-dimensional points (with the one-dimensional points being the leaf nodes).

Algorithm 1 Sphere Decoder

function $\widetilde{\boldsymbol{U}}^*$ = SPHDEC($\widetilde{\boldsymbol{U}}$, $\widetilde{\boldsymbol{U}}_{\text{unc}}$, ρ^2, d^2, i)
  for each $\tilde{u} \in \mathcal{U}$ **do**
    $\widetilde{U}_i \leftarrow \tilde{u}$
    $d'^2 \leftarrow || \widetilde{U}_{\text{unc}_i} - \widetilde{\boldsymbol{H}}_{(i,i:n)} \widetilde{\boldsymbol{U}}_{i:n} ||_2^2 + d^2$
    if $d'^2 \leq \rho^2$ **then**
      if $i > 1$ **then**
        SPHDEC($\widetilde{\boldsymbol{U}}, d'^2, i-1, \rho^2, \widetilde{\boldsymbol{U}}_{\text{unc}}$)
      else
        $\widetilde{\boldsymbol{U}}^* \leftarrow \widetilde{\boldsymbol{U}}$
        $\rho^2 \leftarrow d'^2$
      end if
    end if
  end for
end function

be mentioned is that since $u \in \mathcal{U} = \{-1, 0, 1\}$, i.e., $|\mathcal{U}| = 3$, it is implied that for each node visited during the search process, its two sibling nodes should be also examined in order to get a "certificate" that the optimal solution has been found. If a sibling node is inside the sphere, then further branching occurs and a new node at the next (lower) level of the tree is visited. Otherwise, backtracking happens and previously unexplored nodes at higher dimensions are examined.

The pseudocode of the algorithm presented in [17] is shown in Algorithm 1. The initial values of the arguments are $\widetilde{\boldsymbol{U}} \leftarrow [\,]$, i.e., the empty vector, $\widetilde{\boldsymbol{U}}_{\text{unc}} \leftarrow \widetilde{\boldsymbol{H}}\, \boldsymbol{M}^{-1} \boldsymbol{U}_{\text{unc}}(k)$, $\rho \leftarrow \rho(k)$—see (9), $d \leftarrow 0$, and $i \leftarrow n$.

IV. Alternative Implementation of the Sphere Decoder

Despite the effectiveness of the sphere decoder, as verified in [16] and [17], its computational complexity can be further improved. In the sequel, refinements in the implementation of the algorithm are proposed and the resulting computational complexity—in terms of the real-time operations, as approximated by the floating point operations (flops)—is discussed.

A. Computation of the Initial Radius

As shown in Section III, the initial radius is computed based on the estimated solution $\widetilde{\boldsymbol{U}}_{\text{est}}$ which is equal to either $\widetilde{\boldsymbol{U}}_{\text{bab}}$, or $\widetilde{\boldsymbol{U}}_{\text{ed}}$. Therefore, the initial (squared) radius is

$$\rho^2(k) = || \widetilde{\boldsymbol{U}}_{\text{unc}}(k) - \widetilde{\boldsymbol{H}} \widetilde{\boldsymbol{U}}_{\text{est}}(k) ||_2^2 \,, \tag{13}$$

which can be written, for reasons that will become apparent later, as

$$\rho^2(k) = || \widetilde{\boldsymbol{U}}_{\text{unc}}(k) - (\widetilde{\boldsymbol{H}}_{\text{diag}} + \widetilde{\boldsymbol{H}}_{\text{sut}})\, \widetilde{\boldsymbol{U}}_{\text{est}}(k) ||_2^2 \,, \tag{14}$$

with the diagonal matrix

$$\widetilde{\boldsymbol{H}}_{\text{diag}} = \text{diag}(\tilde{h}_{11}, \tilde{h}_{22}, \ldots, \tilde{h}_{nn}) \,, \tag{15}$$

and the strictly upper triangular matrix

$$\widetilde{\boldsymbol{H}}_{\text{sut}} = \begin{cases} \tilde{h}_{ij} & \text{for } i < j \\ 0 & \text{for } i \geq j \end{cases} \,. \tag{16}$$

By introducing the vector $\boldsymbol{\rho} = [\rho_1 \;\; \rho_2 \;\; \ldots \;\; \rho_n]^T$, with its i^{th} element being the "intermediate" radius ρ_i at level i, $1 \leq i \leq n$, given by[4,5]

$$\begin{aligned} \rho_i^2 &= || \widetilde{\boldsymbol{U}}_{\text{unc}_{i:n}} - (\widetilde{\boldsymbol{H}}_{\text{diag}_{(i:n,i:n)}} + \widetilde{\boldsymbol{H}}_{\text{sut}_{(i:n,i:n)}})\, \widetilde{\boldsymbol{U}}_{\text{est}_{i:n}} ||_2^2 \\ &= (\tilde{u}_{\text{unc}_i} - \underbrace{\tilde{h}_{ii} \tilde{u}_{\text{est}_i}}_{y_{\text{diag}_i}} - \underbrace{\sum_{j=i+1}^{n} \tilde{h}_{ij} \tilde{u}_{\text{est}_j}}_{y_{\text{sut}_i}})^2 + \\ &\underbrace{|| \widetilde{\boldsymbol{U}}_{\text{unc}_{i+1:n}} - (\widetilde{\boldsymbol{H}}_{\text{diag}_{(i+1:n,i+1:n)}} + \widetilde{\boldsymbol{H}}_{\text{sut}_{(i+1:n,i+1:n)}})\, \widetilde{\boldsymbol{U}}_{\text{est}_{i+1:n}} ||_2^2}_{\rho_{i+1}^2} \end{aligned} \tag{17}$$

with $\boldsymbol{y}_{\text{diag}} = \widetilde{\boldsymbol{H}}_{\text{diag}} \widetilde{\boldsymbol{U}}_{\text{est}}$ and $\boldsymbol{y}_{\text{sut}} = \widetilde{\boldsymbol{H}}_{\text{sut}} \widetilde{\boldsymbol{U}}_{\text{est}}$, the initial radius of the hypersphere is then equal to the first element of $\boldsymbol{\rho}$, i.e., $\rho = \rho_1$.

B. Search Process

Before continuing with the search process the following remark is made:

Remark 1: At each level of the search tree at most two sibling nodes can lie within the sphere. Depending on the i^{th} element of the unconstrained solution $\hat{u}_{\text{unc}_i}$ the sibling nodes associated to either the pair of branches $\mathcal{U}_1 = \{0, 1\}$ or $\mathcal{U}_2 = \{-1, 0\}$ can be inside the i^{th} dimension of the sphere.

Proof: Assume that the set of the basis vectors $\tilde{\boldsymbol{h}}_i$ that generate the lattice $\mathcal{L}(\widetilde{\boldsymbol{H}}) = \{\sum_{i=1}^{n} w_i \tilde{\boldsymbol{h}}_i \mid w_i \in \mathbb{Z}\}$ is orthogonal. Moreover, the unconstrained solution $\widetilde{\boldsymbol{U}}_{\text{unc}}(k)$ is at the intersection of the space diagonals of the formed box (n-orthotope) B of side length $\tilde{h}_{ii}$, $1 \leq i \leq n$, and volume $V(B) = \det(\widetilde{\boldsymbol{H}}) = \prod_{i=1}^{n} \tilde{h}_{ii}$. Then—and according to (9)—the smallest hypersphere S that encloses the maximum number of n-dimensional lattice points is the circumscribed hypersphere of the box B, i.e., the sphere that contains B and touches each of its vertices. The n-dimensional bounding sphere S encloses all 2^n vertices of B and its radius ρ_S is

$$\rho_S = \sqrt{\sum_{i=1}^{n} \left(\frac{\tilde{h}_{ii}}{2} \right)^2} \,. \tag{18}$$

Dropping the previous assumption of an orthogonal lattice, let the basis vectors form an arbitrary n-parallelotope B' of volume $V(B') = V(B)$. Then there exists a hypersphere S' centered at the (same) unconstrained solution $\widetilde{\boldsymbol{U}}_{\text{unc}}(k)$ with radius $\rho_{S'} \leq \rho_S$ that does not contain B', but still encloses a nonzero number of its vertices.

To show this, consider a 2-parallelotope (i.e., a parallelogram) spanned by the first two columns of $\widetilde{\boldsymbol{H}}$. The two space diagonals are $\boldsymbol{d}_{B_1'}^{(2)} = \tilde{\boldsymbol{h}}_1 + \tilde{\boldsymbol{h}}_2$ and $\boldsymbol{d}_{B_2'}^{(2)} = \tilde{\boldsymbol{h}}_1 - \tilde{\boldsymbol{h}}_2$, where the superscript within the parentheses denotes the dimension of the

[4] To this end we drop the time-step indication for notational convenience.

[5] Note that the search/computation direction is from the n-dimensional lattice points (i.e., the n^{th} element of the integer-valued vector $\boldsymbol{U}$), to the one-dimensional points (i.e., the first element of $\boldsymbol{U}$).

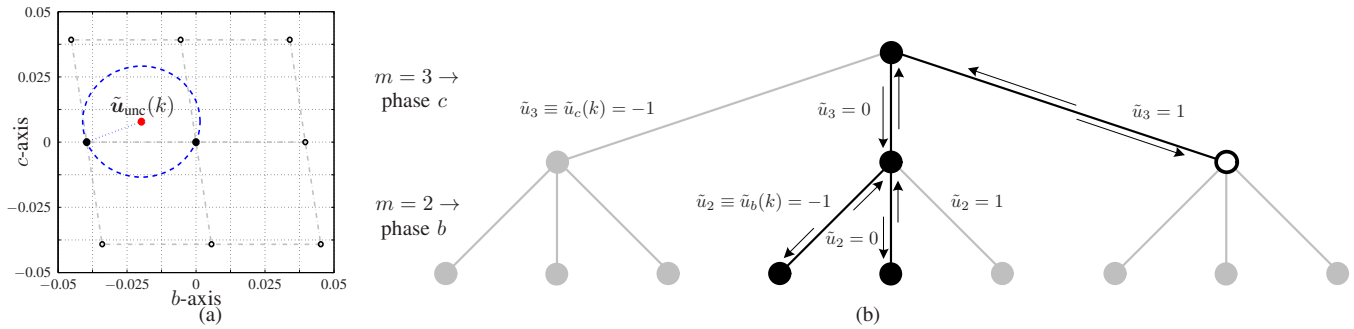


Fig. 3: (a) Sphere decoder in the transformed (two-dimensional) bc-plane as generated by $\tilde{\boldsymbol{H}}$ for the horizon $N = 1$ case. For visualization purposes phase a is not shown. The lattice points are shown as black circles, the unconstrained solution as solid circle, the initial radius and sphere as dotted line and dash-dotted circle, respectively, and the lattice points inside the circle as black solid circles. In this example $\hat{\boldsymbol{U}}_{\text{unc}} = \hat{\boldsymbol{u}}_{\text{unc}} = [-0.4717 \quad 0.2001]^T$. (b) Resulting two-dimensional search tree. Level $m = 3$ corresponds to phase c and level $m = 2$ to phase b. Phase a (i.e., the bottom level of the tree, $m = 1$) is not shown as in Fig. 3(a). The nodes that are evaluated are shown as black solid circles. Those that are examined but do not lie within the sphere are shown as black circles, whereas nodes that are not visited at all are depicted as gray solid circles. The direction of the search process is shown with arrows. Since $\hat{u}_{\text{unc}_c} \equiv \hat{u}_{\text{unc}_3} = 0.2001 \geq 0$, then $\tilde{u}_3 \in \mathcal{U}_1 = \{0, 1\}$. Furthermore, $\hat{u}_{\text{unc}_b} \equiv \hat{u}_{\text{unc}_2} = -0.4717 < 0$ implies that $\tilde{u}_2 \in \mathcal{U}_2 = \{-1, 0\}$.

space. Assuming, without loss of generality, that $\angle(\tilde{\boldsymbol{h}}_1, \tilde{\boldsymbol{h}}_2)$ is not obtuse, then it holds that

$$||\boldsymbol{d}_{B_2'}^{(2)}||_2^2 \leq ||\boldsymbol{d}_B^{(2)}||_2^2 \leq ||\boldsymbol{d}_{B_1'}^{(2)}||_2^2 \Leftrightarrow$$
$$||\tilde{\boldsymbol{h}}_1||_2^2 + ||\tilde{\boldsymbol{h}}_2||_2^2 - 2\tilde{\boldsymbol{h}}_1^T\tilde{\boldsymbol{h}}_2 \leq \tilde{h}_{11}^2 + \tilde{h}_{22}^2 \leq ||\tilde{\boldsymbol{h}}_1||_2^2 + ||\tilde{\boldsymbol{h}}_2||_2^2 + 2\tilde{\boldsymbol{h}}_1^T\tilde{\boldsymbol{h}}_2\,,$$

where $\boldsymbol{d}_B^{(2)}$ is the space diagonal of the two-dimensional rectangular B of same volume (area). Following the same reasoning, for an i-parallelotope there exist two space diagonals $\boldsymbol{d}_{B_1'}^{(i)} = \boldsymbol{d}_{B'}^{(i-1)} + \tilde{\boldsymbol{h}}_i$ and $\boldsymbol{d}_{B_2'}^{(i)} = \boldsymbol{d}_{B'}^{(i-1)} - \tilde{\boldsymbol{h}}_i$, with $\boldsymbol{d}_{B'}^{(i-1)} \leq \boldsymbol{d}_B^{(i-1)}$, such that

$$||\boldsymbol{d}_{B_2'}^{(i)}||_2^2 \leq ||\boldsymbol{d}_B^{(i)}||_2^2 \leq ||\boldsymbol{d}_{B_1'}^{(i)}||_2^2 \Leftrightarrow$$
$$||\boldsymbol{d}_{B'}^{(i-1)}||_2^2 + ||\tilde{\boldsymbol{h}}_i||_2^2 - 2\tilde{\boldsymbol{h}}_i^T\boldsymbol{d}_{B'}^{(i-1)} \leq \sum_{n=1}^{i}\tilde{h}_{ii}^2 \leq ||\boldsymbol{d}_{B'}^{(i-1)}||_2^2 + ||\tilde{\boldsymbol{h}}_i||_2^2 + 2\tilde{\boldsymbol{h}}_i^T\boldsymbol{d}_{B'}^{(i-1)}\,,$$

where, as before, it is assumed that $\angle(\boldsymbol{d}_{B'}^{(i-1)}, \tilde{\boldsymbol{h}}_i) \leq \pi/2$.

Thus, for any n-parallelotope B' there exists at least one space diagonal such that $\boldsymbol{d}_{B'}^{(n)} \leq \boldsymbol{d}_B^{(n)}$. As a result, the hypersphere S' of radius $\rho_{S'} = ||\boldsymbol{d}_{B'}^{(n)}||_2/2$ is not bigger than the hypersphere S of the radius (18). This, also, indicates that ρ_S (see (18)) is an upper bound of $\rho(k)$ in (9).

As can be understood, sphere S' includes at most 2^n lattice points, i.e., at most two lattice points per dimension. This implies that the i^{th} coordinate of the lattice points enclosed by the hypersphere will be either in $\mathcal{U}_1 = \{0, 1\}$ or in $\mathcal{U}_2 = \{-1, 0\}$ depending on where the unconstrained solution lies. If $\hat{u}_{\text{unc}_i} \geq 0$ then the i^{th} coordinate of the lattice points is nonnegative, and if $\hat{u}_{\text{unc}_i} < 0$ it is nonpositive. In any different case, $\boldsymbol{d}_{B'}^{(n)} > \boldsymbol{d}_B^{(n)}$ holds, meaning that a hypersphere of radius greater than ρ_S has been computed, which contradicts the argument above. ■

To provide an intuition on the remark, an illustrative example of a two-dimensional sphere (i.e., a circle) and the resulting search tree is shown in Fig. 3.

The search process in [15] and [17] starts from the root node and moves towards the leaf nodes[6]. In this alternative implementation of the sphere decoder the general search direction remains the same, but with one difference. At the first call of the optimizer, instead of branching from the root node, backtracking occurs since the search process starts from the leaf node $\tilde{u}_{\text{est}_1}$. In particular, since $\widetilde{\boldsymbol{U}}_{\text{est}}$ is, by definition, a feasible solution, it is assumed that this branch has already been fully examined, the bottom level of the search tree has been reached, thus the unexamined sibling nodes (to be more specific, one sibling node, according to Remark 1) of $\tilde{u}_{\text{est}_1}$ have to be visited. This allows one to reduce the computations performed since the optimization stage starts with one candidate solution instead of trying to find one from scratch.

To further reduce the computations by exploiting the structure of the problem (see (17) as well as Remark 1), apart from the aforementioned modification in the starting state of the optimization stage, the search process is divided into two procedures: (a) the branching, and (b) the backtracking procedure. In the following each procedure is described.

1) Backtracking Procedure: When backtracking at level i occurs it means that one of the sibling nodes has already been examined and its associated branch is considered as a tentative element of the candidate solution, denoted with $\tilde{u}_{\text{cnd}_i}$. Therefore, depending on the node already explored associated to $\tilde{u}_{\text{cnd}_i}$ and the value of the unconstrained solution $\hat{u}_{\text{unc}_i}$, the branch $\tilde{u}_i$ to be examined can be (see Fig. 3):

- If $\hat{u}_{\text{unc}_i} \geq 0 \Rightarrow \tilde{u}_{\text{cnd}_i} \in \mathcal{U}_1$. Hence, if $\tilde{u}_{\text{cnd}_i} = 0$ then the sibling node to be examined corresponds to $\tilde{u}_i = 1$, otherwise, if $\tilde{u}_{\text{cnd}_i} = 1$ then the $\tilde{u}_i = 0$ is evaluated.
- If $\hat{u}_{\text{unc}_i} < 0 \Rightarrow \tilde{u}_{\text{cnd}_i} \in \mathcal{U}_2$. Thus, if $\tilde{u}_{\text{cnd}_i} = 0$, then the option $\tilde{u}_i = -1$ is examined, or if $\tilde{u}_{\text{cnd}_i} = -1$ the branch $\tilde{u}_i = 0$ is traversed.

Thus, based on the conditions above, the next branch to be examined is decided. Based on $\tilde{u}_i$ the new "intermediate"

[6]Note, though, that the search tree in these two works is generated in a different way. Specifically, in [15] the lower dimensional points are placed at the higher levels of the tree and the higher dimensional points at the lower levels, whereas in [17] is the other way around, see also footnote 3.

radius ρ_i has to be computed and compared with ρ to find out whether this node lies inside the sphere or not. However, when at level i, the branches from levels n to $i+1$ (i.e., the elements $\{\tilde{u}_{\text{cnd}_n}, \tilde{u}_{\text{cnd}_{n-1}}, \ldots, \tilde{u}_{\text{cnd}_{i+2}}, \tilde{u}_{\text{cnd}_{i+1}}\}$) and the radius ρ_{i+1} are known. Therefore, according to

$$\rho_i^2 = (\tilde{u}_{\text{unc}_i} - \underbrace{\tilde{h}_{ii}\tilde{u}_i}_{y_{\text{diag}_i}} - \underbrace{\sum_{j=i+1}^{n} \tilde{h}_{ij}\tilde{u}_{\text{cnd}_j}}_{y_{\text{sut}_i}})^2 + \underbrace{|| \widetilde{\boldsymbol{U}}_{\text{unc}_{i+1:n}} - (\widetilde{\boldsymbol{H}}_{\text{diag}_{(i+1:n,i+1:n)}} + \widetilde{\boldsymbol{H}}_{\text{sut}_{(i+1:n,i+1:n)}})\, \widetilde{\boldsymbol{U}}_{\text{cnd}_{i+1:n}} ||_2^2}_{\rho_{i+1}^2}, \tag{19}$$

y_{sut_i} is also known, meaning that the only term that has to be computed is the y_{diag_i}. Note that the above expression is the same as (17) with the difference that the tentative elements of the candidate solution are taken into account since, as the process progresses, they replace those of the estimated solution. Of course, if $\rho_i \leq \rho$ then branching is done (see next subsection), the i^{th} element of $\boldsymbol{\rho}$ and $\widetilde{\boldsymbol{U}}_{\text{cnd}}$ are updated, and the process moves to the $i-1$ level of the tree. Otherwise, backtracking occurs and the $i+1$ level is visited.

2) Branching Procedure: The principle of the branching procedure is the same as that in [17], with the difference that—and in accordance with Remark 1, see also Fig. 3—only two sibling nodes are examined in the worst case. Again, the criterion for choosing which pair of nodes is eventually examined is based on the value of the unconstrained solution $\hat{u}_{\text{unc}_i}$. Before starting exploring the branches, though, and by considering that the process moved from level $i+1$ to level i, the i^{th} *tentative* y_{sut_i} has to be computed to save computations. Upon doing so, the branches of the i^{th} level are examined. Thereby, if

- $\hat{u}_{\text{unc}_i} \geq 0$ then the nodes associated to the branches $\tilde{u}_i \in \mathcal{U}_1 = \{0, 1\}$ are visited. In particular, first the branch $\tilde{u}_i = 0$ is traversed and if $\rho_i|_{\tilde{u}_i=0} \leq \rho$ then branching is done. If $\rho_i|_{\tilde{u}_i=0} > \rho$, then this branch is pruned and the branch $\tilde{u}_i = 1$ is followed and tested whether $\rho_i|_{\tilde{u}_i=1} \leq \rho$ holds. If it does, then $\tilde{u}_i = 1$ is the tentative element of the candidate solution and the search proceeds to the $i-1$ level of the tree. If, on the other hand, $\rho_i|_{\tilde{u}_i=1} > \rho$ then both sibling nodes are considered examined, no branching can be done, thus the process has reached a dead-end, and backtracking occurs.
- $\hat{u}_{\text{unc}_i} < 0$ then the branches to be considered are $\tilde{u}_i \in \mathcal{U}_2 = \{-1, 0\}$. The same logic is followed as when $\tilde{u}_i \in \mathcal{U}_1$, with the first branch traversed being the $\tilde{u}_i = 0$, and in case this does not result in a smaller radius then $\tilde{u}_i = -1$ is checked.

Finally, it should be mentioned that when branching is done, the i^{th} element of $\boldsymbol{y}_{\text{sut}}$, $\boldsymbol{\rho}$ and $\widetilde{\boldsymbol{U}}_{\text{cnd}}$ are updated accordingly. Furthermore, when a leaf node is reached the radius of the sphere is updated $\rho \leftarrow \rho_1$ and a candidate solution has been found.

Algorithm 2 Alternative Sphere Decoder

function $\widetilde{\boldsymbol{U}}^*$ = SPHDECALT($\widetilde{\boldsymbol{U}}_{\text{est}}, \widetilde{\boldsymbol{U}}_{\text{unc}}, \hat{\boldsymbol{U}}_{\text{unc}}, \boldsymbol{\rho}, \rho^2, \boldsymbol{y}_{\text{sut}}, i$)
  $\widetilde{\boldsymbol{U}}_{\text{cnd}} \leftarrow \widetilde{\boldsymbol{U}}_{\text{est}}$, $\boldsymbol{\rho}_{\text{tmp}} \leftarrow \boldsymbol{\rho}$
  if solutionFound **then**
    $\widetilde{\boldsymbol{U}}^* \leftarrow \widetilde{\boldsymbol{U}}_{\text{cnd}}$, $\rho \leftarrow \rho_1$
  end if
  for each $i \in \{1, \ldots, n\}$ **do**
    if backtracking **then**
      if $\hat{u}_{\text{unc}_i} \geq 0$ **then**
        $\tilde{u}_i \in \mathcal{U}_1 \setminus \tilde{u}_{\text{cnd}_i}$
      else
        $\tilde{u}_i \in \mathcal{U}_2 \setminus \tilde{u}_{\text{cnd}_i}$
      end if
      $\rho^2_{\text{tmp}_i} \leftarrow (\tilde{u}_{\text{unc}_i} - \tilde{h}_{ii}\tilde{u}_i - y_{\text{sut}_i})^2 + \rho^2_{i+1}$
      COMPANDUPD($\rho^2_{\text{tmp}_i}, \rho^2, i$)
    else if branching **then**
      $y_{\text{sut}_i} \leftarrow \widetilde{\boldsymbol{H}}_{\text{sut}_{(i,i+1:n)}} \widetilde{\boldsymbol{U}}_{\text{cnd}_{(i+1:n)}}$
      if $\hat{u}_{\text{unc}_i} \geq 0$ **then**
        $\tilde{u}_i \in \mathcal{U}_1$
      else
        $\tilde{u}_i \in \mathcal{U}_2$
      end if
      for each $\tilde{u}_i$ **do**
        $\rho^2_{\text{tmp}_i} \leftarrow (\tilde{u}_{\text{unc}_i} - \tilde{h}_{ii}\tilde{u}_i - y_{\text{sut}_i})^2 + \rho^2_{i+1}$
        COMPANDUPD($\rho^2_{\text{tmp}_i}, \rho^2, i$)
      end for
    end if
  end for
end function

function $[\tilde{u}_{\text{CND}_i}, \rho_i^2, i]$ = COMPANDUPD($\rho^2_{\text{tmp}_i}$, ρ^2, i)
  if $\rho^2_{\text{tmp}_i} \leq \rho^2$ **then**
    $\tilde{u}_{\text{cnd}_i} \leftarrow \tilde{u}_i$, $\rho_i^2 \leftarrow \rho^2_{\text{tmp}_i}$
    if $i = 1$ **then**
      backtracking && solutionFound $\leftarrow$ true
      $i \leftarrow i + 1$
    else
      branching $\leftarrow$ true
      $i \leftarrow i - 1$
    end if
  else
    backtracking $\leftarrow$ true
    $i \leftarrow i + 1$
  end if
end function

The pseudocode of the alternative implementation of the sphere decoder is presented in Algorithm 2. The initial values of the arguments are $\widetilde{\boldsymbol{U}}_{\text{est}} \leftarrow \widetilde{\boldsymbol{U}}_{\text{bab}}(k)$ or $\widetilde{\boldsymbol{U}}_{\text{ed}}(k)$ depending on (9), $\widetilde{\boldsymbol{U}}_{\text{unc}} \leftarrow \widetilde{\boldsymbol{H}}\, \boldsymbol{M}^{-1}\boldsymbol{U}_{\text{unc}}(k)$, $\hat{\boldsymbol{U}}_{\text{unc}} \leftarrow \boldsymbol{M}^{-1}\boldsymbol{U}_{\text{unc}}(k)$, $\boldsymbol{\rho} \leftarrow [\rho_1(k)\ \rho_2(k)\ \ldots\ \rho_n(k)]^T$, $\rho \leftarrow \rho_1(k)$, $\boldsymbol{y}_{\text{sut}} \leftarrow \widetilde{\boldsymbol{H}}_{\text{sut}}\, \widetilde{\boldsymbol{U}}_{\text{est}}(k)$, and $i \leftarrow 1$. Note that to guarantee termination of the algorithm, a stopping criterion can be implemented, as proposed in [21].

C. Computational Complexity

Based on the previous discussion and (19), when backtracking occurs only two additions (one for $y_{\text{diag}_i} + y_{\text{sut}_i}$ and the other for $(*)^2 + \rho_{i+1}^2$), one multiplication (for squaring the term within the parentheses $(*)^2$) and one subtraction are required, regardless of the dimension of the lattice point. This is in contrast to [17], where for an i-dimensional point $n - i + 2$ additions (since the complete dot product $\widetilde{\boldsymbol{H}}_{(i,i:n)}\widetilde{\boldsymbol{U}}_{\,i:n}$ is computed, meaning that $n-i$ additions are required in addition to these two mentioned before), one multiplication and one subtraction are required. Moreover, this is done only for one sibling node and not for two as in Algorithm 1.

With regards to the computations performed during the branching procedure, these are summarized as follows. For the computation of the tentative y_{sut_i}, performed once per level, $n - i$ additions are required for an i-dimensional lattice point. Then for each point, regardless of its dimension, only two additions (one for $y_{\text{diag}_i} + y_{\text{sut}_i}$ and the other for $(*)^2 + \rho_{i+1}^2$), one multiplication (for squaring the term within the parentheses $(*)^2$) and one subtraction are required. Finally, it should be noted that—as with the backtracking procedure—at most two sibling nodes are examined.

Summarizing the computations performed, the following expressions result

$$\begin{aligned} N_a &= 2(\mu - 1) + \sum_{\nu=1}^{\mu} \big(n - m(\nu)\big)\,, \\ N_s &= 2\mu\,, \quad N_m = 2\mu\,, \end{aligned} \tag{20}$$

where $1 \leq m \leq n$ denotes the dimension of the point (i.e., level of the tree), and μ the maximum number of nodes visited. Furthermore, N_a, N_s and N_m denote the number of additions, subtractions and multiplications, respectively. As can be seen, the reduction in the flop count is significant compared to Algorithm 1, where the respective flops are given by

$$\begin{aligned} N_a &= 3\left(\mu - 1 + \sum_{\nu=1}^{\mu} \big(n - m(\nu)\big)\right)\,, \\ N_s &= 3\mu\,, \quad N_m = 3\mu\,. \end{aligned} \tag{21}$$

Note that the same number of nodes is visited with both implementations[7]. What changes is the flop number.

In Fig. 4 the computational complexity—in terms of the real-time flops—is shown. More specifically, the maximum number of operations performed N_t by the presented algorithm is shown for different prediction horizons and it is compared with that of the sphere decoder as proposed in [17] and of its initial implementation [15]. Note that (slightly) more nodes are visited with the latter approach. As can be seen, for a ten-step horizon the flops are reduced by up to 55% and 75% compared to the approaches proposed in [17] and [15], respectively[8].

[7]Strictly speaking, this is not totally true. As explained, with the proposed implementation n less nodes are examined thanks to the fact the search process starts from a leaf node, after having a candidate (i.e., the estimated) solution.

[8]If we do not consider the n nodes visited when computing the initial radius, then the reduction is greater. For example, for the ten-step horizon case, the flops performed are 3,254, i.e., the reduction in the flops reaches 61% and 79% compared to the approaches proposed in [17] and [15].

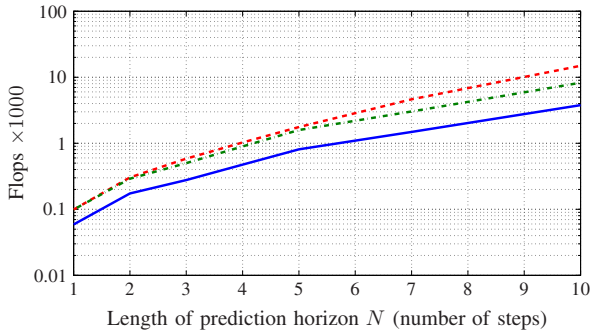


Fig. 4: Maximum number of flops N_t as a function of the prediction horizon N as resulting from the sphere decoder in [15] (dashed line), the sphere decoder in [17] (dash-dotted line), and the proposed algorithm (solid line).

V. Performance Evaluation

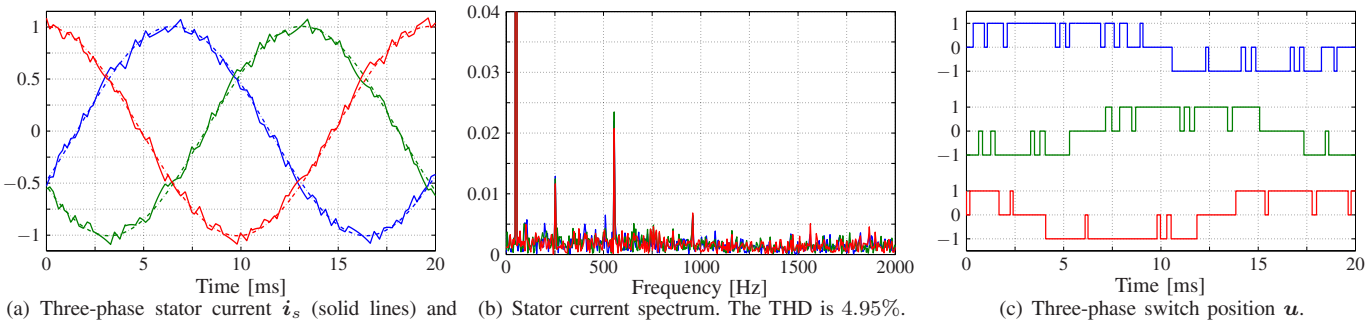
The presented simulation results were obtained based on an MV drive (Fig. 1) consisting of a squirrel cage IM with 3.3 kV rated voltage, 356 A rated current, 2 MVA rated power, 50 Hz nominal frequency, 0.25 p.u. total leakage inductance, and a three-level NPC with the constant dc-link voltage $V_{\text{dc}} = 5.2$ kV and a fixed neutral point N. The sampling interval was set to $T_s = 25\,\mu$s. For all cases examined below, the converter was operated at a switching frequency of approximately 300 Hz by appropriately tuning λ_u. All results are shown in the p.u. system.

The steady-state performance of the drive is shown in Fig. 5. The horizon $N = 10$ case is investigated; the weighting factor $\lambda_u = 0.1$ is chosen. As can be seen in Fig. 5(a), the three-phase stator current waveforms—illustrated over one fundamental period—accurately track their references. The resulting current spectrum is shown in Fig. 5(b); the total harmonic distortion (THD)—which quantifies the accuracy performance of the controller—is relatively low (4.95%), considering the low switching frequency. Finally, Fig. 5(c) shows the three-phase switching sequence.

In Fig. 6 the THD for different lengths of the prediction horizon is shown. As can be seen the THD decreases as the horizon increases, as expected. What it is worthwhile to mention, is that the exact same drive performance is achieved with the algorithms presented in [15] and [17]. This implies that the proposed modifications do not affect the optimization stage, meaning that the optimal solution is always found.

VI. Conclusions

This paper proposes an alternative implementation of the sphere decoding algorithm employed to solve the long-horizon direct model predictive control (MPC) problem for current reference tracking. The optimization algorithm is implemented in a way to fully exploit the structure of the problem; both the branching and backtracking procedures are refined to keep the real-time computations to a minimum, and at the same time without sacrificing optimality. Thanks to the proposed modifications, the computational burden can be reduced by more than 75% and 55% for long horizons and a three-level converter, compared to that required for the search algorithms proposed in [15] and [17], respectively.



(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 $\boldsymbol{u}$.

Fig. 5: Simulated waveforms produced by the direct model predictive control problem with current reference tracking at steady-state operation, at full speed and rated torque. A ten-step horizon ($N = 10$) is used, the sampling interval is $T_s = 25\,\mu$s and the weighting factor $\lambda_u = 0.1$ such that a switching frequency of approximately 300 Hz results.

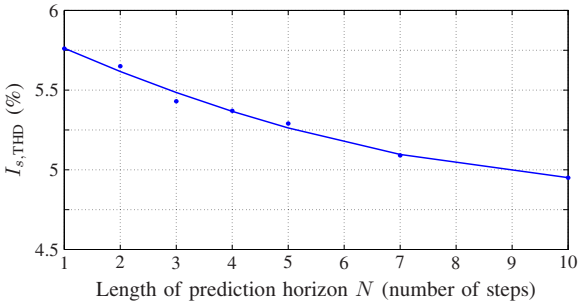


Fig. 6: Stator current THD as a function of the prediction horizon N. The data points relate to individual simulations, which were approximated using a polynomial function of second order.

Appendix

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

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

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

References

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

[2] 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.

[3] S. Vazquez, J. I. Leon, L. G. Franquelo, J. Rodríguez, H. A. Young, A. Marquez, and P. Zanchetta, "Model predictive control: A review of its applications in power electronics," *IEEE Ind. Electron. Mag.*, vol. 8, no. 1, pp. 16–31, Mar. 2014.

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

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

[13] P. Karamanakos, T. Geyer, and R. Kennel, "Computationally efficient optimization algorithms for model predictive control of linear systems with integer inputs," in *Proc. IEEE Conf. Decis. Control*, Osaka, Japan, Dec. 2015, pp. 3663–3668.

[14] ——, "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.

[15] 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.

[16] ——, "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.

[17] 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.

[18] R. P. Aguilera, R. Baidya, P. Acuna, S. Vazquez, T. Mouton, and V. G. Agelidis, "Model predictive control of cascaded H-bridge inverters based on a fast-optimization algorithm," in *Proc. IEEE Ind. Electron. Conf.*, Yokohama, Japan, Nov. 2015, pp. 4003–4008.

[19] 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.

[20] 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.

[21] 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.

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

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