A Computationally Efficient Model Predictive Control Strategy for Linear Systems With Integer Inputs

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

***Abstract*—For linear systems with integer inputs, the model predictive control (MPC) problem with output reference tracking is formulated as an integer least-squares (ILS) problem. The ILS problem is solved using a modified sphere decoding algorithm, which is a particular branch-and-bound method. To reduce the computational complexity of the sphere decoder, a reduction algorithm is added as a preprocessing stage to reshape the search space in which the integer solution lies. The computational complexity of the proposed algorithm is modest, enabling its implementation in a real-time system even when considering long prediction horizons. A variable speed drive system with a three-level voltage source inverter serves as an illustrative example to demonstrate the effectiveness of the proposed algorithm.**

***Index Terms*—Model predictive control (MPC), integer least-squares (ILS) problem, integer programming, LLL lattice basis reduction, sphere decoding, power electronics, drive systems**

I. Introduction

THROUGHOUT the years, model predictive control (MPC) [1] has gained popularity in many disciplines due to its numerous advantages, including its ability to handle systems with complex dynamics, such as hybrid systems. For linear systems with integer inputs, some (or all) of the decision variables of the MPC problem are integer-valued, and the optimization problem underlying MPC is a (mixed) integer program [2], [3], which is NP-hard.This type of systems can be modeled as mixed logical dynamical (MLD) systems [4], hybrid automata [5], or polyhedral piecewise affine systems [6].

The explicit state-feedback control law can be computed offline for such systems [7], [8], or they can be solved online, using for example branch-and-bound methods [3]. The former approach, however, typically requires significant memory resources to store the explicit control law, with the required memory increasing dramatically with the problem size and complexity. Explicit control laws are also ill-suited to address variations in parameters or reference setpoints.

For long prediction horizons or for problems with many integer variables, due to the combinatorial explosion of the number of possible integer solutions, solving the integer optimization problem online might lead to computational intractability [9]. Long horizons are often required to ensure stability and good closed-loop performance [10], [11], as was shown for different applications in various fields [12]–[15]. Short sampling intervals further aggravate this issue. In the field of power electronics, for example, sampling intervals of hundreds or even tens of microseconds are common.

To address this issue, a dedicated optimization technique was recently proposed in [16], [17] for MPC problems involving linear systems with integer inputs. The underlying optimization problem is formulated as an integer least-squares (ILS) problem and the branch-and-bound technique of sphere decoding [18], [19] is adopted to compute the optimal sequence of control actions.

By reformulating the ILS problem through a preprocessing stage, [20] proposed a modified version of the algorithm introduced in [16]. A lattice reduction algorithm [21] was implemented to reduce the size of the n-dimensional search space and, as a result, the number of nodes in the search tree to be examined. In addition, the implementation of the sphere decoder in [16] was refined. With these improvements, the computations to be performed in real time can be reduced, while still obtaining the optimal solution.

In this work, the initial results presented in [20] are extended. The formulation of the MPC problem is generalized and applied to general linear systems with integer inputs. An exhaustive description of the derivation and solution process of the reformulated ILS problem is provided, along with a detailed analysis of the complexity of the proposed algorithm.

To provide further insight, an example of a linear system with integer inputs from the field of power electronics is used as a case study. More specifically, a variable speed drive system is considered, which consists of a three-level neutral point clamped (NPC) inverter driving a medium-voltage induction machine. For long horizons, such as ten steps, the computational complexity of the discussed strategy is reduced by up to 45% compared with the approach presented in [16], highlighting the efficacy of the proposed algorithm.

II. Formulation of the Optimal Control Problem

Consider a linear system with integer inputs that is described by the discrete-time state-space model

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k)\,,\ \boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k)\,, \qquad (1)$$

which has n_x states, n_y outputs and n_u inputs. In this formulation, $\boldsymbol{x} \in \mathbb{R}^{n_x}$ is the state vector, $\boldsymbol{y} \in \mathbb{R}^{n_y}$ the output vector, and $\boldsymbol{u} \in \boldsymbol{\mathcal{U}}$ is the integer-valued input vector, 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

$\boldsymbol{\mathcal{U}} = \mathcal{U} \times \cdots \times \mathcal{U} = \mathcal{U}^{n_u}$ being the n_u-times Cartesian product of the set $\mathcal{U} \subset \mathbb{Z}$. The state-space matrices $\boldsymbol{A} \in \mathbb{R}^{n_x \times n_x}$, $\boldsymbol{B} \in \mathbb{R}^{n_x \times n_u}$ and $\boldsymbol{C} \in \mathbb{R}^{n_y \times n_x}$ are time invariant, even though the proposed control approach is also applicable to time-varying matrices. Finally, $k \in \mathbb{N}$ denotes the time step.

A. Model Predictive Control with Output Reference Tracking

Consider MPC with output reference tracking as depicted in Fig. 1 with the objective function

$$J(k) = \sum_{\ell=k}^{k+N-1} ||\boldsymbol{y}_{\text{err}}(\ell+1)||^2_{\boldsymbol{Q}} + ||\Delta \boldsymbol{u}(\ell)||^2_{\boldsymbol{R}} \tag{2}$$

that penalizes over the finite prediction horizon N the evolution of the output error $\boldsymbol{y}_{\text{err}}(k) = \boldsymbol{y}_{\text{ref}}(k) - \boldsymbol{y}(k)$, and the control effort $\Delta \boldsymbol{u}(k) = \boldsymbol{u}(k) - \boldsymbol{u}(k-1)$. The ratio between the weighting matrices $\boldsymbol{Q} \in \mathbb{R}^{n_y \times n_y}, \boldsymbol{Q} \succeq 0$ and $\boldsymbol{R} \in \mathbb{R}^{n_u \times n_u}, \boldsymbol{R} \succ 0$ decides on the trade-off between the overall tracking accuracy and the control effort.

At time step k, the optimal solution (i.e., the control input) is obtained by solving the following problem in real time

$$\begin{aligned} \underset{\boldsymbol{U}(k)}{\text{minimize}} \quad & J(k) \\ \text{subject to} \quad & \boldsymbol{x}(\ell+1) = \boldsymbol{A}\boldsymbol{x}(\ell) + \boldsymbol{B}\boldsymbol{u}(\ell) \\ & \boldsymbol{y}(\ell) = \boldsymbol{C}\boldsymbol{x}(\ell)\,,\ \forall\, \ell = k, \dots, k+N-1 \\ & \boldsymbol{U}(k) \in \mathbb{U}\,. \end{aligned} \tag{3}$$

The optimization variable is the integer-valued sequence of control actions $\boldsymbol{U}(k) = [\boldsymbol{u}^T(k)\ \dots\ \boldsymbol{u}^T(k+N-1)]^T \in \mathbb{U} = \boldsymbol{\mathcal{U}}^N \subset \mathbb{Z}^n$ over the N-step horizon, where $n = N \cdot n_u$. It should be mentioned that because of the nature of system (1) a steady-state error may be present[1].

B. Formulation of the ILS Problem

Using algebraic manipulations, the optimization problem (3) can be reformulated as an ILS problem. To this end, let the vectors $\boldsymbol{Y}(k) = [\boldsymbol{y}^T(k+1)\ \dots\ \boldsymbol{y}^T(k+N)]^T$ and $\boldsymbol{Y}_{\text{ref}}(k) = [\boldsymbol{y}^T_{\text{ref}}(k+1)\ \dots\ \boldsymbol{y}^T_{\text{ref}}(k+N)]^T$ denote the sequence of outputs and the corresponding sequence of output references over the horizon, respectively.

The objective function J can be written in vector form as

$$\begin{aligned} J(k) = ||\boldsymbol{\Gamma}\boldsymbol{x}(k) + \boldsymbol{\Upsilon}\boldsymbol{U}(k) - \boldsymbol{Y}_{\text{ref}}(k)||^2_{\tilde{\boldsymbol{Q}}} + \\ + ||\boldsymbol{S}\boldsymbol{U}(k) - \boldsymbol{\Xi}\boldsymbol{u}(k-1)||^2_{\tilde{\boldsymbol{R}}}\,, \end{aligned} \tag{4}$$

with $\boldsymbol{Y}(k) = \boldsymbol{\Gamma}\boldsymbol{x}(k) + \boldsymbol{\Upsilon}\boldsymbol{U}(k)$. The matrices $\boldsymbol{\Gamma}$, $\boldsymbol{\Upsilon}$, $\boldsymbol{S}$ and $\boldsymbol{\Xi}$ can be found in [20], whereas the block diagonal matrices $\tilde{\boldsymbol{Q}}$, $\tilde{\boldsymbol{R}}$ are defined as $\tilde{\boldsymbol{Q}} = \oplus_{i=1}^N \boldsymbol{Q}$, and $\tilde{\boldsymbol{R}} = \oplus_{i=1}^N \boldsymbol{R}$. After some algebraic manipulations (as explained in the appendix) the objective function (4) can be written as

$$J(k) = \big(\boldsymbol{U}(k) - \boldsymbol{U}_{\text{unc}}(k)\big)^T \boldsymbol{W} \big(\boldsymbol{U}(k) - \boldsymbol{U}_{\text{unc}}(k)\big) + \text{const}(k)\,, \tag{5}$$

where $\boldsymbol{U}_{\text{unc}} \in \mathbb{R}^n$ is the unconstrained solution of the problem (3), see the appendix. The matrix $\boldsymbol{W}$ is by definition symmetric positive definite and can thus be factored (using Cholesky factorization) as $\boldsymbol{W} = \boldsymbol{H}^T\boldsymbol{H}$, where $\boldsymbol{H} \in \mathbb{R}^{n \times n}$ is

[1]Note that no stability guarantees are provided in this work.

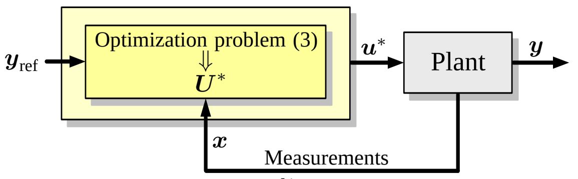


Fig. 1: Model predictive control with output reference tracking.

a nonsingular, upper triangular matrix. As a result, problem (3) becomes the ILS problem

$$\underset{\boldsymbol{U}(k) \in \mathbb{U}}{\text{minimize}} \quad ||\bar{\boldsymbol{U}}_{\text{unc}}(k) - \boldsymbol{H}\boldsymbol{U}(k)||^2_2\,, \tag{6}$$

where $\bar{\boldsymbol{U}}_{\text{unc}} = \boldsymbol{H}\boldsymbol{U}_{\text{unc}}(k) \in \mathbb{R}^n$. The matrix $\boldsymbol{H}$ is commonly known as the lattice generator matrix—its columns represent n linearly independent vectors in $\mathbb{R}^n$ that generate a lattice, i.e., the set of integer linear combinations of the basis vectors $\boldsymbol{h}_i$, i.e., $\mathcal{L}(\boldsymbol{H}) = \{\sum_{i=1}^n \kappa_i \boldsymbol{h}_i \mid \kappa_i \in \mathbb{Z}\}$.

III. Solving the ILS Problem

To facilitate the real-time implementation of the proposed MPC algorithm the ILS problem (6) needs to be solved in a time-efficient manner. To achieve this, the procedure for solving the problem is divided into two stages: the first stage (preprocessing) consists of the *reduction* of problem (6), whereas in the second stage (optimization) the *search* for the optimal solution is performed.

A. Preprocessing Stage

In a first step, the Lenstra-Lenstra-Lovász (LLL) lattice basis reduction algorithm, as proposed in [21], is applied to the problem (6). The goal is to improve the conditioning of the optimization problem (6) by transforming $\boldsymbol{H}$ to a new upper triangular matrix $\tilde{\boldsymbol{H}} \in \mathbb{R}^{n \times n}$. To do so, the reduction process—which can be accomplished with only a few computations, thanks to the sparsity of $\boldsymbol{H}$—produces the matrix $\tilde{\boldsymbol{H}}$ with positive diagonal entries that satisfy the following criteria:

$$|\tilde{h}_{i,j}| \leq \frac{1}{2}\tilde{h}_{i,i}, \quad i = 1, \dots, j-1 \tag{7a}$$

$$\delta \tilde{h}^2_{j-1,j-1} \leq \tilde{h}^2_{j-1,j} + \tilde{h}^2_{j,j}, \quad j = 2, \dots, n\,, \tag{7b}$$

where $1/4 < \delta \leq 1$ (with the typical value being $\delta = 3/4$).

To this end, integer Gauss transformations (IGTs) and permutation matrices are employed. First, IGTs of the form $\boldsymbol{M}_{i,j} = \boldsymbol{I} - \gamma \boldsymbol{e}_i \boldsymbol{e}_j^T$ are applied to the right-hand side of $\boldsymbol{H}$, where $\boldsymbol{e}_i$ is the unit vector (i.e., the ith column of the identity matrix $\boldsymbol{I}$ of proper dimensions) and $\gamma \in \mathbb{Z}$. Therefore, by post-multiplying $\boldsymbol{H}$ with $\boldsymbol{M}_{i,j}$ (with $i < j$) we obtain

$$\tilde{\boldsymbol{H}} = \boldsymbol{H}\boldsymbol{M}_{i,j} = \boldsymbol{H} - \gamma \boldsymbol{H}\boldsymbol{e}_i\boldsymbol{e}_j^T\,. \tag{8}$$

The matrices $\boldsymbol{H}$ and $\tilde{\boldsymbol{H}}$ are the same except for the (k,i) entries, with $k = 1, 2, \dots, i$, which are $\tilde{h}_{k,j} = h_{i,j} - \gamma h_{k,i}$. By following a procedure similar to Gaussian elimination, i.e., by setting[2] $\gamma = \lfloor h_{i,j}/h_{i,i} \rceil$, we ensure that the entry $|\tilde{h}_{i,j}|$ is sufficiently reduced such that it meets condition (7a).

[2]$\lfloor \xi \rceil$ denotes the integer that is obtained by rounding ξ. This definition can be directly extended to vectors $\boldsymbol{\xi}$, by performing elementwise rounding.

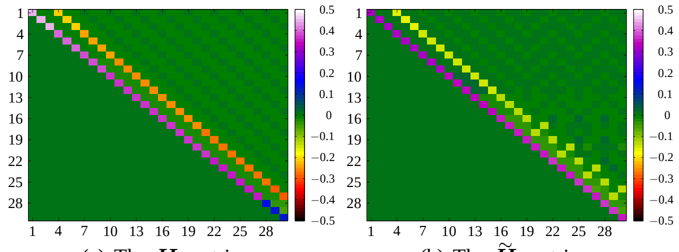


Fig. 2: Visualization of the lattice generator matrix (a) before and (b) after the LLL reduction algorithm, assuming the dimension $n = 30$. The diagonal entries of the upper triangular matrix $\widetilde{\boldsymbol{H}}$ are positive and are placed in an ascending order. Moreover, their values are more than two times larger than those of the off-diagonal entries, thus conditions (7a) and (7b) hold.

To satisfy the second condition (7b), permutations of the entries of $\boldsymbol{H}$ are required whenever $\delta h^2_{j-1,j-1} > h^2_{j-1,j} + h^2_{j,j}$. A Givens rotation matrix $\boldsymbol{G}_{j-1,j}$ [22] is employed to maintain the upper triangular form of the resulting matrix $\widetilde{\boldsymbol{H}}$. Hence, the LLL reduced matrix $\widetilde{\boldsymbol{H}}$ can be written as

$$\widetilde{\boldsymbol{H}} = \boldsymbol{G}^T_{j-1,j}\boldsymbol{H}\boldsymbol{P}_{j-1,j}\,, \tag{9}$$

where the Givens $\boldsymbol{G}_{j-1,j}$ and permutation $\boldsymbol{P}_{j-1,j}$ matrices are described in [20].

With this procedure, the second condition, as described by (7b), is met, since

$$\begin{cases} \tilde{h}_{j-1,j-1} = \sqrt{h^2_{j-1,j} + h^2_{j,j}} \\ \tilde{h}_{j-1,j} = c h_{j-1,j-1}\,,\ \ c = h_{j-1,j}(h^2_{j-1,j} + h^2_{j,j})^{-1/2} \\ \tilde{h}_{j,j} = -s h_{j-1,j-1}\,,\ \ s = h_{j,j}(h^2_{j-1,j} + h^2_{j,j})^{-1/2} \end{cases} .$$

As a result, the reduction process strives to place the diagonal entries of $\widetilde{\boldsymbol{H}}$ in ascending order, i.e.,

$$\tilde{h}_{1,1} \leq \tilde{h}_{2,2} \leq \ldots \leq \tilde{h}_{n,n}\,. \tag{10}$$

Note that the optimization algorithm described in the next section performs a depth-first search. Ordering the diagonal entries of $\widetilde{\boldsymbol{H}}$ reduces the number of nodes to be considered in the early steps of the search procedure, thus decreasing the total number of nodes to be explored.

The pseudocode of the LLL reduction procedure is provided in Algorithm 1. An illustrative example[3] of the effect of the LLL algorithm on $\boldsymbol{H}$ is presented in Fig. 2. More formally, $\boldsymbol{H}$ is transformed into $\widetilde{\boldsymbol{H}}$, which is of the form

$$\widetilde{\boldsymbol{H}} = \boldsymbol{V}^T\boldsymbol{H}\boldsymbol{M}\,, \tag{11}$$

with $\boldsymbol{V} \in \mathbb{R}^{n\times n}$ being an orthogonal matrix and $\boldsymbol{M} \in \mathbb{Z}^{n\times n}$ being unimodular (i.e., $\det \boldsymbol{M} = \pm 1$). With this, the ILS problem (6) can be rewritten as

$$\underset{\widetilde{\boldsymbol{U}}(k)\,\in\,\mathbb{U}}{\text{minimize}}\ \ ||\,\widetilde{\boldsymbol{U}}_{\text{unc}}(k) - \widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{U}}(k)||^2_2\,, \tag{12}$$

with $\widetilde{\boldsymbol{U}}_{\text{unc}}(k) = \widetilde{\boldsymbol{H}}\,\boldsymbol{M}^{-1}\boldsymbol{U}_{\text{unc}}(k)$ and $\widetilde{\boldsymbol{U}}(k) = \boldsymbol{M}^{-1}\boldsymbol{U}(k)$.

[3]The entries of the lattice generator matrices $\boldsymbol{H}$ and $\widetilde{\boldsymbol{H}}$ are computed based on the case study presented in Section IV. As explained there, the example is based on a prediction horizon of $N = 10$ steps.

Algorithm 1 LLL Reduction

```
function $\widetilde{H}$ = LLL($H$)
    $M \leftarrow I_n$
    while $j \leftarrow 2 \leq n$ do
        if $|h_{j-1,j}| > \frac{1}{2}|h_{j-1,j-1}|$ then
            $H \leftarrow HM_{j-1,j}$ , $M \leftarrow MM_{j-1,j}$
        end if
        if $\delta h^2_{j-1,j-1} > h^2_{j-1,j} + h^2_{j,j}$ then
            $H \leftarrow G^T_{j-1,j}HP_{j-1,j}$ , $M \leftarrow MP_{j-1,j}$
            $j \leftarrow \max\{j-1, 2\}$
        else
            for $i \leftarrow j-2$ down to $1$ do
                if $|h_{i,j}| > \frac{1}{2}|h_{i,i}|$ then
                    $H \leftarrow HM_{i,j}$ , $M \leftarrow MM_{i,j}$
                end if
            end for
            $j \leftarrow j+1$
        end if
    end while
    $\widetilde{H} \leftarrow H$
end function
```

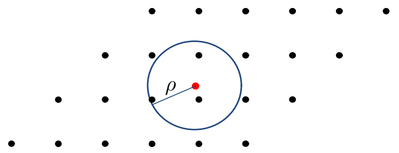


Fig. 3: The principle of the sphere decoder: A (two-dimensional) sphere of radius ρ (shown as blue line) centered at the unconstrained solution (shown as red solid circle) includes several integer points of the lattice (shown as black solid circles). One of these points is the integer solution of the ILS problem.

B. Optimization Stage

To find the optimal solution in the transformed lattice generated by $\widetilde{\boldsymbol{H}}$, a sphere decoding algorithm is implemented, based on the one described in [16]. This algorithm is a variant of the Fincke and Pohst sphere decoding algorithm [18], which exploits the fact that the set of admissible control input sequences $\mathbb{U}$ forms a (truncated) integer lattice. According to the sphere decoding principle, the optimal solution lies within a hypersphere (n-dimensional sphere) of radius ρ, see Fig. 3. As mentioned in Section III-A, the sphere decoding algorithm is a depth-first search algorithm that finds the optimal solution by traversing a search tree in a sequential manner until reaching a dead end or the bottom level. If this happens, the algorithm backtracks to examine unexplored nodes in higher layers. An example of an integer search tree is provided in Fig. 4 along with the visualization of the search procedure.

To exploit the structure of $\widetilde{\boldsymbol{H}}$ and to avoid unnecessary computations, the search tree is built in a bottom-to-top manner with the higher-dimensional nodes being located at the top layers of the search tree, and the one-dimensional nodes at the bottom, see Fig. 4. This is in contrast to [16], where the tree is generated in a top-to-bottom manner. The pseudocode of the proposed algorithm is summarized in Algorithm 2, where the initial values of the arguments are $\widetilde{\boldsymbol{U}} \leftarrow [\ ]$, i.e., the empty vector, $d \leftarrow 0$, $i \leftarrow n$, $\rho \leftarrow \rho(k)$—see (15), and $\widetilde{\boldsymbol{U}}_{\text{unc}} \leftarrow \widetilde{\boldsymbol{H}}\,\boldsymbol{M}^{-1}\boldsymbol{U}_{\text{unc}}(k)$.

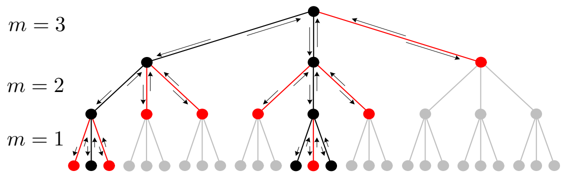


Fig. 4: Search tree of the ILS problem (12) of dimension (depth) $n = 3$. The integer set $\mathcal{U}$ has cardinality three. The index $m = \{1, \ldots, n\}$ refers to the layer in the search tree. As the search progresses, the dimension of the sphere is reduced from $m = n$ to a one-dimensional sphere. The nodes explored by the sphere decoder are shown as black solid circles. Those that do not lie within the 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.

Before invoking Algorithm 2, the initial radius $\rho(k)$ of the hypersphere needs to be determined. A common choice for $\rho(k)$ is the so-called Babai estimate [23], which is a suboptimal solution to the ILS problem[4]. For the ILS problem (6), the Babai estimate can be easily computed by rounding the unconstrained solution to the closest *feasible* integer vector

$$\boldsymbol{U}_{\text{bab}}(k) = \lfloor \boldsymbol{H}^{-1}\bar{\boldsymbol{U}}_{\text{unc}}(k) \rceil = \lfloor \boldsymbol{U}_{\text{unc}}(k) \rceil \in \mathbb{U}\,, \tag{13}$$

which is equivalent [24] to

$$\begin{aligned} \epsilon_n &= \bar{u}_{\text{unc}_n}/h_{n,n} \Rightarrow u_{\text{bab}_n} = \lfloor \epsilon_n \rceil\,, \\ \epsilon_i &= \Big(\bar{u}_{\text{unc}_i} - \sum_{j=i+1}^{n} h_{i,j} u_{\text{bab}_j}\Big)/h_{i,i} \Rightarrow u_{\text{bab}_i} = \lfloor \epsilon_i \rceil\,. \end{aligned} \tag{14}$$

The initial value of $\rho(k)$ is then

$$\rho(k) = ||\,\widetilde{\boldsymbol{U}}_{\text{unc}}(k) - \widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{U}}_{\text{bab}}(k)||_2\,, \tag{15}$$

where $\widetilde{\boldsymbol{U}}_{\text{bab}}(k) = \boldsymbol{M}^{-1}\boldsymbol{U}_{\text{bab}}(k)$, and thus it results that [19]

$$\rho^2(k) \geq \sum_{i=1}^{n} \Big(\tilde{u}_{\text{unc}_i} - \sum_{j=1}^{n} \tilde{h}_{i,j}\tilde{u}_j\Big)^2\,. \tag{16}$$

Therefore, the n conditions to be met so that the solution set is nonempty and at least one lattice point exists are[5] [19]

$$\begin{aligned} \left\lceil \frac{-\rho + \tilde{u}_{\text{unc}_n}}{\tilde{h}_{n,n}} \right\rceil &\leq \tilde{u}_n \leq \left\lfloor \frac{\rho + \tilde{u}_{\text{unc}_n}}{\tilde{h}_{n,n}} \right\rfloor\,, \\ \left\lceil \frac{-\rho_{n-1} + \tilde{u}_{\text{unc}_{n-1|n}}}{\tilde{h}_{n-1,n-1}} \right\rceil &\leq \tilde{u}_{n-1} \leq \left\lfloor \frac{\rho_{n-1} + \tilde{u}_{\text{unc}_{n-1|n}}}{\tilde{h}_{n-1,n-1}} \right\rfloor\,, \end{aligned} \tag{17}$$

where the conditions for the $(n-2)$-dimensional node $\tilde{u}_{n-2}$ up to the one-dimensional node $\tilde{u}_1$ can be derived by continuing the above-shown procedure in a similar fashion. Note that in (17), $\rho_{n-1} = \big(\rho^2 - (\tilde{u}_{\text{unc}_n} - \tilde{h}_{n,n}\tilde{u}_n)^2\big)^{1/2}$ and $\tilde{u}_{\text{unc}_{n-1|n}} = \tilde{u}_{\text{unc}_{n-1}} - \tilde{h}_{n-1,n}\tilde{u}_n$.

Finally, it should be pointed out that when the LLL reduction algorithm is used to transform the lattice, the probability

[4]In [16] an alternative calculation method for $\rho(k)$ is proposed, which exploits the receding horizon policy of MPC. More specifically, the previously computed optimal control sequence is shifted by one step back in time.

[5]$\lceil \xi \rceil$ maps ξ to the smallest following integer (the smallest integer not less than ξ). Conversely, $\lfloor \xi \rfloor$ maps ξ to the largest previous integer (the largest integer not greater than ξ).

Algorithm 2 Sphere Decoder

```
function $\widetilde{\boldsymbol{U}}^*$ = SPHDEC($\widetilde{\boldsymbol{U}}$, $d^2$, $i$, $\rho^2$, $\widetilde{\boldsymbol{U}}_{\text{unc}}$)
    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
```

of the Babai estimate $\boldsymbol{U}_{\text{bab}}$ being equal to the optimal solution $\boldsymbol{U}^*$ is increased, as shown in [24]. As a consequence, in most cases the sphere that includes at least one lattice point has the smallest possible radius ρ, and thus the nodes to be explored by the algorithm are reduced.

IV. Case Study: Three-Level NPC Inverter Drive System

To highlight the efficacy and the benefits of the proposed algorithm to solve the ILS problem in the general form (6), an industrial case study is presented in this section. More specifically, a medium-voltage drive system is considered, which is based on a three-level NPC voltage source inverter, as shown in Fig. 5. The inverter uses the constant dc-link voltage $V_{\text{dc}} = 5.2\,\text{kV}$ and has a fixed neutral point potential. A squirrel cage induction machine is connected to the inverter with $3.3\,\text{kV}$ rated voltage, $356\,\text{A}$ rated current, $2\,\text{MVA}$ rated power, $50\,\text{Hz}$ nominal frequency, $596\,\text{rpm}$ nominal rotational speed and a 0.25 per unit (p.u.) total leakage inductance. For all cases examined, the control algorithm is operated with the sampling interval $T_s = 25\,\mu\text{s}$.

A. Mathematical Model of the System

The inverter produces at each phase the voltages $-\frac{V_{\text{dc}}}{2}, 0, \frac{V_{\text{dc}}}{2}$, depending on the position of the corresponding semiconductor switches. The switch positions in the phase legs can be described by the integer variables $u_a, u_b, u_c \in \mathcal{U} = \{-1, 0, 1\}$; these variables constitute the manipulated variables, which are aggregated to the three-phase vector $\boldsymbol{u} = [u_a\ u_b\ u_c]^T \in \boldsymbol{\mathcal{U}} = \mathcal{U}^3$. The state vector $\boldsymbol{x} = [i_{s\alpha}\ i_{s\beta}\ \psi_{r\alpha}\ \psi_{r\beta}]^T \in \mathbb{R}^4$ encompasses the stator current $\boldsymbol{i}_{s,\alpha\beta}$ and the rotor flux $\boldsymbol{\psi}_{r,\alpha\beta}$ in the $\alpha\beta$ plane[6]. By choosing the stator current as the output of the system, i.e., $\boldsymbol{y} = \boldsymbol{i}_{s,\alpha\beta} \in \mathbb{R}^2$, the discrete-time state-space model[7] of the drive system can be written in the form (1). Note that in the examined case study, the system dimensions are given by $n_x = 4$, $n_y = 2$ and $n_u = 3$ according to the notation introduced in Section II.

[6]Note that to simplify the computations it is common to transform a vector from the three-phase (abc) to the stationary orthogonal $\alpha\beta$ system.

[7]For the continuous-time state equations and for the detailed derivation of the system mathematical model see [25] and [16], respectively.

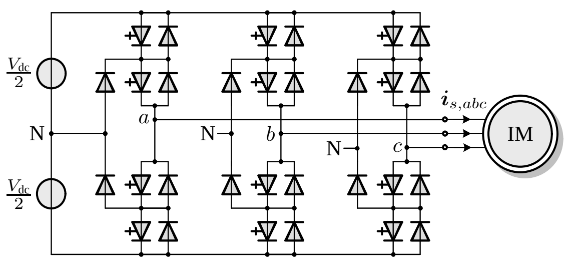


Fig. 5: Three-level three-phase neutral point clamped (NPC) voltage source inverter driving an induction machine with a fixed neutral point potential.

B. Direct MPC with Current Reference Tracking

The block diagram of the MPC scheme with current reference tracking is alike the one in Fig. 1, with the feedback signals being the (measured) stator current and rotor speed, and the (estimated) rotor flux. The main control objective is the elimination of the current error $\boldsymbol{i}_{s,\text{err},\alpha\beta}$ [20]. To do so, the switches of the inverter are *directly* manipulated without the presence of a modulator. Moreover, since for medium-voltage drives switching losses are of paramount importance, the switching (i.e., control) effort is to be kept small.

Considering the aforementioned objectives, we set in (2) $\boldsymbol{y}_{\text{err}} = \boldsymbol{i}_{s,\text{err},\alpha\beta}$. Moreover, since the $\alpha-$ and $\beta-$components of the stator current are of the same magnitude, we chose $\boldsymbol{Q} = \boldsymbol{I}$ and $\boldsymbol{R} = \lambda_u \boldsymbol{I}$, where $\lambda_u > 0$ sets the trade-off between the stator current tracking accuracy and the switching effort (i.e., the switching frequency).

V. Analysis of the Computational Complexity

The computational complexity of the proposed MPC algorithm is analyzed in this section, using the drive system as a case study. Since the system matrices are time invariant for a given speed setpoint and dc-link voltage, $\boldsymbol{W}$ and thus the lattice generator matrix $\boldsymbol{H}$ are time invariant, too. As a result, the preprocessing stage can be performed offline and only the sphere decoding algorithm needs to be executed in real time. The latter thus determines the online computational demand, while the first part of the algorithm is given for reasons of completeness.

This motivates us to divide the analysis of the computational burden into two parts: (a) the preprocessing stage (Algorithm 1), and (b) the optimization stage (Algorithm 2). The focus of this analysis is on the worst-case number of operations, which is, from an implementation point of view, the decisive quantity rather than the average number of operations.

A. Complexity of the Preprocessing Stage

For the LLL reduction algorithm, as introduced in [21], the average complexity is $O(n^4 \log n)$ floating-point operations (flops)[8]. However, as shown in [26], the average complexity can be reduced to $O(n^3 \log n)$, assuming that the entries of the lattice generator matrix $\boldsymbol{H}$ are independent and identically distributed (i.i.d.) random variables $H \sim \mathcal{N}(0,1)$. In our case, however, the entries of $\boldsymbol{H}$ are highly correlated and $\boldsymbol{H}$ is sparse. Therefore, the computational complexity is expected to be less than $O(n^3 \log n)$, since the vast majority of the off-diagonal entries of $\boldsymbol{H}$ already satisfy criteria (7a) and (7b). This becomes also evident from the example shown in Fig. 2.

[8]The notion of flop count is used to provide a rough estimate of the computation time of the implemented algorithm. It is not an accurate estimate, especially when n is small ($n < 100$), since the computation time on today's platforms highly depends on the architecture and compiler.

TABLE I: Maximum number of nodes μ evaluated and flops N_t performed in real time by the sphere decoder without and with the LLL reduction algorithm, depending on the length of the prediction horizon N.

Prediction	Without LLL		With LLL	
horizon N	$\max(\mu)$	$\max(N_t)$	$\max(\mu)$	$\max(N_t)$
1	7	99	7	99
2	19	369	14	291
3	39	915	19	501
4	87	2,905	27	897
5	148	5,667	44	1,587
7	690	10,275	61	3,030
10	831	42,147	141	8,268

B. Complexity of the Sphere Decoding Algorithm

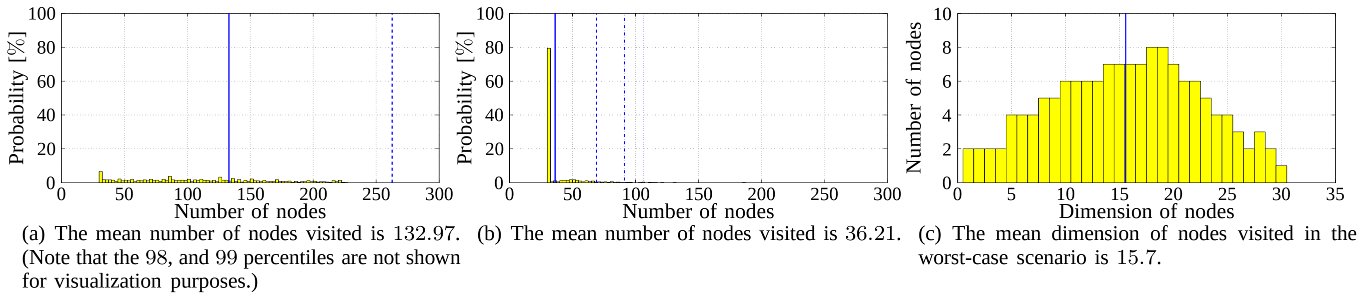
The ILS problem is known to be NP-hard [27], [28]. In order to derive an upper bound on the required computations, an empirical analysis is presented hereafter. To do so, the number of nodes that are evaluated by the sphere decoder at each time-step to obtain the solution is computed. More specifically, the maximum number of nodes μ examined is investigated for lattices of different dimensions (i.e., for different prediction horizons, since the length of the horizon defines the dimension of the lattice, and vice versa).

The lattice generator matrix $\boldsymbol{H}$ is computed for the drive system case study detailed in the previous section. The weighting factor λ_u is tuned such that a switching frequency of about $300\,\text{Hz}$ results.

Table I summarizes the computational burden of the sphere decoder in terms of the number of nodes examined. Two cases are examined: the case without the LLL reduction algorithm (i.e., $\boldsymbol{H}$ is the lattice generator matrix), and the case with the LLL reduction algorithm (i.e., $\widetilde{\boldsymbol{H}}$ is the lattice generator matrix). As can be seen, the sphere decoder with the LLL algorithm significantly reduces the number of nodes to be examined.

Figs. 6(a) and 6(b) illustrate the probability distribution of the number of nodes required to be explored at each time-step to find the optimal solution, for the ten-step horizon ($N = 10$) case. When the ILS problem is ill-conditioned the distribution is relatively flat (with a low peak at 30), spread over a wide range of numbers (from 30 up to 225), see Fig. 6(a). On the other hand, when the search space is reshaped and the ILS problem is transformed to a well-conditioned one, the probability distribution exhibits a large peak at 30. With 30 being the depth of the search tree, this number constitutes the minimum number of nodes to be explored by the algorithm. More precisely, as can be observed in Fig. 6(b), in about 80% of the cases, the solution is found in the first iteration.

Even though the complexity analysis based on the number of nodes provides a first indication, it is rather coarse. The number of computations depends not only on the number of nodes explored, but also on the dimension of these nodes, see Fig. 4. As an example, consider after the preprocessing stage



(a) The mean number of nodes visited is 132.97. (Note that the 98, and 99 percentiles are not shown for visualization purposes.) (b) The mean number of nodes visited is 36.21. (c) The mean dimension of nodes visited in the worst-case scenario is 15.7.

Fig. 6: (a) and (b): Probability distribution of the number of nodes visited by the sphere decoding algorithm (Algorithm 2), for a ten-step horizon ($N = 10$) (a) before, and (b) after the LLL reduction algorithm. The mean number of nodes visited is indicated by the solid vertical line. The 95 and 98, and 99 percentiles are shown as dashed, dash-dotted, and dotted vertical lines, respectively. (c): Dimension of each node visited in the worst-case scenario by the sphere decoder when using a ten-step horizon and the preprocessing stage. The mean dimension of the nodes is indicated by the solid vertical line.

sphere decoding for the horizon $N = 10$ case with the worst-case scenario ($\max(\mu) = 141$ nodes visited). The number of nodes explored at each layer is shown in Fig. 6(c). As can be seen, many nodes are of high dimensions ($m > 15$), implying that few flops are required. This is due to the use of an upper triangular $\widetilde{\boldsymbol{H}}$ matrix and the effective pruning of suboptimal branches.

In the sequel, a second and more precise analysis of the computational burden is performed for the real-time operations. Since the computation of the unconstrained solution $\widetilde{\boldsymbol{U}}_{\text{unc}}$ is performed in real time, the corresponding operations are to be taken into account. Considering that $\widetilde{\boldsymbol{U}}_{\text{unc}} = \widetilde{\boldsymbol{H}}\, \boldsymbol{M}^{-1}\boldsymbol{U}_{\text{unc}}$, and that $\widetilde{\boldsymbol{H}}$ is upper triangular, whereas $\boldsymbol{M}$ is sparse, then $n(n+1)/2$ multiplications and $n(n-1)/2$ additions are required, i.e., n^2 flops in total.

The operations performed by the sphere decoder are a function of the dimension of the node examined. Focusing on a snapshot of the recursive Algorithm 2, it can be concluded that the most effort is required to compute the update of the radius of the hypersphere. Specifically, for an m-dimensional node, $n - m + 1$ additions[9] ($n - m$ for the computation of $\widetilde{\boldsymbol{H}}_{(m,m:n)}\widetilde{\boldsymbol{U}}_{m:n}$ and one for the addition $|| * ||_2^2 + d^2$), one subtraction, and $n - m + 2$ multiplications ($n - m + 1$ for the computation of $\widetilde{\boldsymbol{H}}_{(m,m:n)}\widetilde{\boldsymbol{U}}_{m:n}$ and one for squaring the Euclidean norm $|| * ||_2^2$) are performed. More precisely, since $\boldsymbol{U} \in \mathbb{U}$, the result of the $n - m + 1$ multiplications performed at the mth layer is either the multiplicand itself, with the same or reversed sign, or zero. Therefore, only one multiplication is performed at each node, regardless of its dimension. Finally, since the cardinality of $\mathcal{U}$ is three for the case study, for each child node explored, the two sibling nodes are evaluated to determine whether they lie inside of the hypersphere. Note that divisions are not required.

Taking all the above into account, the total number of additions N_a, subtractions N_s and multiplications N_m performed in real time is

$$\begin{aligned} N_a &= \frac{n(n-1)}{2} + 3\left(\mu - 1 + \sum_{\nu=1}^{\mu}\big(n - m(\nu)\big)\right), \\ N_s &= 3\mu\,, \\ N_m &= \frac{n(n+1)}{2} + 3\mu\,. \end{aligned} \tag{18}$$

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

To derive an upper bound on the number of operations to be performed, a worst-case scenario is examined. This corresponds to the case where (a) the maximum number of nodes $\max(\mu)$ is explored by the sphere decoder (see Table I), and (b) $\boldsymbol{U} \in \mathbb{U} \setminus \{0\}$, since multiplications with zero result in a decrease in the number of additions.

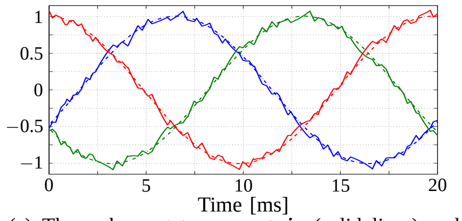
The total number of operations $N_t = N_a + N_s + N_m$ corresponds to the flop count for the real-time computation. The upper bound on the operations performed in real time $\max(N_t)$, is also summarized in Table I. For the horizon $N = 10$ case, the worst case number of flops is reduced by 80%, compared with a reduction of 83% when considering the worst case number of nodes.

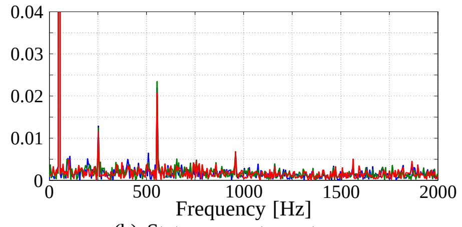
VI. Performance Evaluation

The simulation results presented in this section relate to the medium-voltage drive system depicted in Fig. 5. The horizon $N = 10$ case is investigated. The weighting factor $\lambda_u = 0.1$ is chosen, such that a switching frequency—corresponding to the control effort—of approximately $300\,\text{Hz}$ is obtained.

The steady-state performance of the drive is shown in Fig. 7. The normalized three-phase stator current waveforms and their references are illustrated over one fundamental period in Fig. 7(a). In Fig. 7(b) the resulting current spectrum is depicted to visualize the tracking accuracy of the controller. The total harmonic distortion (THD) of the current is used as a performance metric, which quantifies the tracking performance of the controller. The current THD is with 4.95% relatively low given the low switching frequency, as can also be seen in [17] where the direct MPC current controller with one- and multiple-step horizon is compared with the pulsewidth modulation (PWM) based strategy of space vector modulation (SVM), and a synchronous optimal modulation technique, namely the offline-calculated optimized pulse patterns (OPPs). The good performance of the controller can also be validated by the thorough comparison presented in [29] between several MPC and PWM-based techniques. Finally, Fig. 7(c) depicts the control input over one fundamental period, i.e., the three-phase switching sequence.

Interestingly, in the field of power electronics, the ILS problem underlying direct MPC is typically solved using a brute-force approach, namely exhaustive enumeration of all admissible sequences of control inputs [30]. This limits the prediction horizon achievable in a real-time implementation


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


(b) Stator current spectrum.

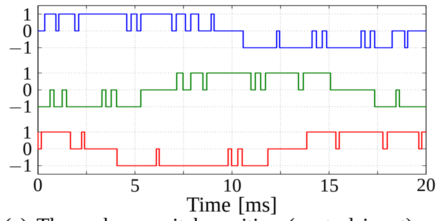

(c) Three-phase switch position (control input) $\boldsymbol{u}$.

Fig. 7: Simulated waveforms produced by the direct model predictive controller 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 is $\lambda_u = 0.1$. The switching frequency (interpreted as the control effort) is approximately $300\,$Hz and the current THD (interpreted as the controller tracking accuracy) is 4.95%.

TABLE II: Maximum number of nodes μ evaluated and flops N_t performed by (a) the exhaustive search algorithm, (b) the sphere decoder in [16], and (c) the proposed algorithm. The resulting current THD for each prediction horizon is also shown.

Prediction	Exhaustive search		Sphere decoder [16]		Proposed approach		
horizon N	$\max(\mu)$	$\max(N_t)$	$\max(\mu)$	$\max(N_t)$	$\max(\mu)$	$\max(N_t)$	THD %
1	26	250	7	99	7	99	5.76
2	517	4,948	16	304	14	291	5.65
3	7,371	70,120	24	585	19	501	5.43
4	103,518	983,587	31	1,029	27	897	5.37
5	1,455,000	$> 1 \cdot 10^7$	50	1,764	44	1,587	5.29
7	$> 3 \cdot 10^8$	$> 5 \cdot 10^9$	99	4,635	61	3,030	5.09
10	$> 2 \cdot 10^{12}$	$> 4 \cdot 10^{13}$	255	14,936	141	8,268	4.95

typically to one [31], despite several strategies to facilitate the implementation of MPC algorithms with nontrivial prediction horizons [32]. Recently, the concept of sphere decoding has been introduced to the power electronics community in [16].

The computational complexity of the proposed approach is compared in Table II with these two approaches from the literature. Specifically, the table shows the maximum number of examined nodes μ and operations performed N_t for different prediction horizons. As before, the converter operates at a switching frequency of about $300\,$Hz, regardless of the prediction horizon. The resulting current THD is shown to highlight that the closed-loop performance of the system can be improved by using longer prediction horizons.

Exhaustive search clearly becomes impractical for prediction horizons exceeding two steps. When the sphere decoder is used to restrict the search space the computational complexity—even in the worst-case scenario—increases only mildly as the prediction horizon is extended. Compared to the approach proposed in [16], the sphere decoding algorithm with the modifications proposed in this work further reduces both the number of nodes examined and flops performed. The maximum number of nodes is reduced by about 45%, and that of the operations performed in real time by a very similar percentage, i.e., by 44%.

This reduction is a result of the reshaping of the search space and the initial radius ρ being smaller, as can be seen in Fig. 8(a). The computation of the initial radius based on the Babai estimate (see (13) and (14)) results in a slightly tighter hypersphere, and consequently reduces the number of nodes explored. This can be observed in Figs. 8(b) and 8(c), where the probability distributions of the number of nodes visited by the sphere decoder are illustrated. As can be seen, the shape of the distributions resulting from the two sphere decoding approaches is very similar. Yet, with the proposed algorithm, it is more concentrated at the lower end of 30 nodes.

VII. Conclusions

The optimization problem underlying model predictive control (MPC) of linear systems with integer inputs is an integer least-squares (ILS) problem. This paper discusses a computationally efficient method to solve this problem, consisting of two steps. First, in a preprocessing step, a lattice reduction algorithm transforms the optimization problem into a well-conditioned problem, such that it can be solved more efficiently. These computations are performed once and offline.

Second, in the optimization stage, a refined sphere decoding algorithm is used to solve the optimization problem in a branch-and-bound manner and to derive the optimal sequence of control inputs. The initial radius of the hypersphere is computed using the Babai estimate, ensuring that the solution set is always nonempty while the radius is as small as possible. Nevertheless, since real-time guarantees of termination cannot be provided, modifications—such as adding a stopping criterion—are required to resolve this issue. This matter is further discussed in [33].

An industrial case study of a three-level inverter drive system is used to illustrate the benefits of the optimization method. The computational complexity is analyzed—both in terms of the number of nodes in the search tree explored and the number of flops required. Compared to the optimization method proposed in [16], the computational burden is reduced by up to 45%.

Appendix

The objective function (4) can be written as

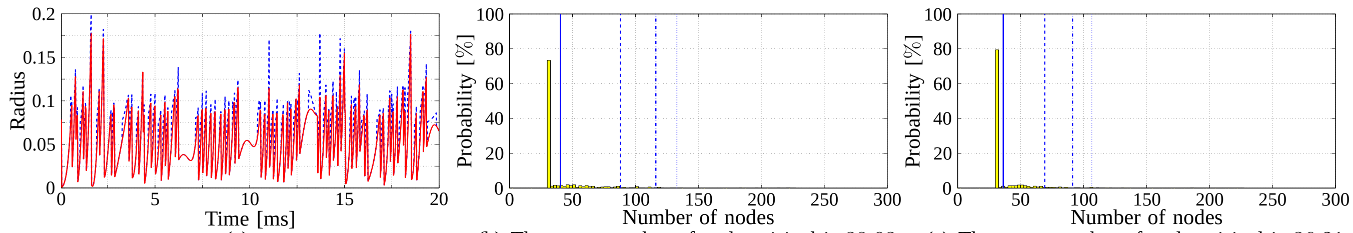
$$J(k) = \zeta(k) + ||\boldsymbol{U}(k)||^2_{\boldsymbol{W}} + \boldsymbol{U}^T(k)\boldsymbol{\Lambda}(k) + \boldsymbol{\Lambda}^T(k)\boldsymbol{U}(k)\,, \quad (19)$$

where

$$\zeta(k) = ||\boldsymbol{\Gamma}\boldsymbol{x}(k) - \boldsymbol{Y}_{\text{ref}}(k)||^2_{\tilde{\boldsymbol{Q}}} + ||\boldsymbol{\Xi}\boldsymbol{u}(k-1)||^2_{\tilde{\boldsymbol{R}}}\,,$$

$$\boldsymbol{W} = \boldsymbol{\Upsilon}^T\,\tilde{\boldsymbol{Q}}\,\boldsymbol{\Upsilon} + \boldsymbol{S}^T\,\tilde{\boldsymbol{R}}\,\boldsymbol{S}\,,$$

$$\boldsymbol{\Lambda}(k) = \Big(\big(\boldsymbol{\Gamma}\boldsymbol{x}(k) - \boldsymbol{Y}_{\text{ref}}(k)\big)^T\,\tilde{\boldsymbol{Q}}\,\boldsymbol{\Upsilon} - \big(\boldsymbol{\Xi}\boldsymbol{u}(k-1)\big)^T\,\tilde{\boldsymbol{R}}\,\boldsymbol{S}\Big)^T$$



(a) (b) The mean number of nodes visited is 38.93. (c) The mean number of nodes visited is 36.21.

Fig. 8: (a): Initial radius ρ as a function of time, based on the Babai estimate (13) (red solid line) and the educated guess proposed in [16] (blue dashed line). (b) and (c): Probability distribution of the number of nodes visited by the sphere decoding algorithm, for a ten-step horizon ($N = 10$), as proposed (b) in [16], and (c) in this work. The mean number of nodes visited is indicated by the solid vertical line. The 95 and 98, and 99 percentiles are shown as dashed, dash-dotted, and dotted vertical lines, respectively.

Since $\boldsymbol{R} \succ 0 \Rightarrow \boldsymbol{W} \succ 0$, and $\boldsymbol{W} = \boldsymbol{W}^T$, after rearranging terms and completing the squares, (19) can be written as

$$J(k) = ||\boldsymbol{U}(k)+\boldsymbol{W}^{-1}\boldsymbol{\Lambda}(k)||^2_{\boldsymbol{W}} + \underbrace{\zeta(k) - \boldsymbol{\Lambda}^T(k)\boldsymbol{W}^{-T}\boldsymbol{\Lambda}(k)}_{\text{const}(k)} \,. \tag{20}$$

Note that the constant term is independent of $\boldsymbol{U}(k)$, and thus it can be neglected. Allowing $\boldsymbol{U}(k) \subset \mathbb{R}^n$, the unconstrained (i.e., relaxed) solution of (20) is $\boldsymbol{U}_{\text{unc}}(k) = -\boldsymbol{W}^{-1}\boldsymbol{\Lambda}(k)$. Inserting $\boldsymbol{U}_{\text{unc}}(k)$ in (20), (5) results, which in turn, becomes the ILS function in problem (6). ■

References

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

[2] C. A. Floudas, *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications*. Oxford, UK: Oxford Univ. Press, 1995.

[3] L. A. Wolsey, *Integer Programming*. New York, NY: Wiley, 1998.

[4] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, Mar. 1999.

[5] R. Alur, C. Courcoubetis, T. A. Henzinger, and P. H. Ho, "Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems," in *Hybrid Syst.*, ser. LNCS, R. Grossman, A. Nerode, A. Ravn, and H. Rischel, Eds. Springer-Verlag, 1993, vol. 736, pp. 209–229.

[6] E. D. Sontag, "Nonlinear regulation: The piecewise linear approach," *IEEE Trans. Autom. Control*, vol. 26, no. 2, pp. 346–358, Apr. 1981.

[7] F. Borrelli, M. Baotić, A. Bemporad, and M. Morari, "Dynamic programming for constrained optimal control of discrete-time linear hybrid systems," *Automatica*, vol. 41, no. 10, pp. 1709–1721, Oct. 2005.

[8] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge, UK: Cambridge Univ. Press, 2011.

[9] D. E. Quevedo, G. C. Goodwin, and J. A. De Doná, "Finite constraint set receding horizon quadratic control," *Int. J. of Robust Nonlin. Control*, vol. 14, no. 4, pp. 355–377, Mar. 2004.

[10] M. Morari and J. H. Lee, "Model predictive control: Past, present and future," *Comput. and Chemical Eng.*, vol. 23, no. 4, pp. 667–682, May 1999.

[11] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, Jun. 2000.

[12] G. C. Goodwin, H. Haimovich, D. E. Quevedo, and J. S. Welsh, "A moving horizon approach to networked control system design," *IEEE Trans. Autom. Control*, vol. 49, no. 9, pp. 1427–1445, Sep. 2004.

[13] D. E. Quevedo and G. C. Goodwin, "Multistep optimal analog-to-digital conversion," *IEEE Trans. Circuits Syst. I*, vol. 52, no. 3, pp. 503–515, Mar. 2005.

[14] ——, "Moving horizon design of discrete coefficient FIR filters," *IEEE Trans. Signal Process.*, vol. 53, no. 6, pp. 2262–2267, Jun. 2005.

[15] D. E. Quevedo, J. Østergaard, and D. Nešić, "Packetized predictive control of stochastic systems over bit-rate limited channels with packet loss," *IEEE Trans. Autom. Control*, vol. 56, no. 12, pp. 2854–2868, Dec. 2011.

[16] 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.

[17] ——, "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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] J. W. Demmel, *Applied Numerical Linear Algebra*. Philadelphia, PA: SIAM, 1997.

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

[24] 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.

[25] 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.

[26] C. Ling, W. H. Mow, and N. Howgrave-Graham, "Reduced and fixed-complexity variants of the LLL algorithm for communications," *IEEE Trans. Commun.*, vol. 61, no. 3, pp. 1040–1050, Mar. 2013.

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

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

[29] T. Geyer, "A comparison of control and modulation schemes for medium-voltage drives: Emerging predictive control concepts versus PWM-based schemes," *IEEE Trans. Ind. Appl.*, vol. 47, no. 3, pp. 1380–1389, May/Jun. 2011.

[30] J. Rodríguez, J. Pontt, C. A. Silva, P. Correa, P. Lezana, P. Cortés, and U. Ammann, "Predictive current control of a voltage source inverter," *IEEE Trans. Ind. Electron.*, vol. 54, no. 1, pp. 495–503, Feb. 2007.

[31] 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.

[32] 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.

[33] 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.