Computationally efficient optimization algorithms for model predictive control of linear systems with integer inputs

Computationally Efficient Optimization Algorithms for Model Predictive Control of Linear Systems With Integer Inputs

Petros Karamanakos, Tobias Geyer and Ralph Kennel

Abstract— **The model predictive control problem of linear systems with integer inputs results in an integer optimization problem. In case of a quadratic objective function, the optimization problem can be cast as an integer least-squares (ILS) problem. Three algorithms to solve this problem are proposed in this paper. Optimality can be traded in to reduce the computation time. An industrial case study—an inverter-driven electrical drive system—is considered to demonstrate the effectiveness of the presented techniques.**

I. Introduction

Model predictive control (MPC) [1] is well-suited to control systems with complex dynamics, such as hybrid systems. When linear systems with integer inputs are concerned, some (or all) of the decision variables of the underlying optimization problem are integers, meaning that the problem is a (mixed) integer program ((M)IP) [2], which is NP-hard.

Solving (M)IPs in real time is—in general—challenging. With the help of convex relaxations [3], MIPs can be cast as convex optimization problems [4]. This leads, however, in general to suboptimal solutions and entails a certain performance degradation. To find the exact solution, it is common practice to either compute the explicit control law [5] or to use branch-and-bound methods [2]. Despite the advantage that the explicit control law can be computed offline, the required memory to store it is often prohibitive for problems of a meaningful size. System parameters and time-varying references increase the parameter vector and thus the memory requirement.

Branch-and-bound methods are sensitive to the starting condition, rely on smart branching heuristics and require effective bounding criteria to prune suboptimal branches. In some cases, tight lower bounds can be produced with quadratic (QP) and semidifinite programming (SDP) [6]–[8] relaxations on the integer problem [9]. Implementing an efficient branch-and-bound technique allows one to solve the (M)IP in real time, while keeping the computational complexity at bay, thus enabling the implementation of long-horizon MPC schemes that are often required to ensure stability and good closed-loop performance [10].

This paper focuses on computationally efficient algorithms to solve the (M)IP underlying MPC of linear systems with integer inputs. First, an optimization algorithm called sphere decoding is presented that yields the optimal solution [11], [12]. This branch-and-bound optimization technique, which was recently proposed in [13], [14] to solve MPC problems of the type considered here, can be employed when the optimization problem is formulated as an integer least-squares (ILS) problem. As a result of its bounding strategy, the optimal sequence of control actions is found with only a (relatively) small number of operations.

Furthermore, two solution techniques are proposed that exhibit shorter computation times, albeit at the expense of optimality. With regards to the first approach, the initial guess is always implemented, without triggering the optimization process at all. For computing it, only a quadratic amount of real-time operations is required. However, optimality is sacrificed (not significantly, though, as shown) since in some cases it is a suboptimal solution. According to the second algorithm, suboptimal solutions are only occasionally implemented depending on whether a predefined upper bound on the floating-point operations (flops) performed in real time is hit, or not. In that way, the computation time is bounded, whereas the plant performance is kept close to the optimal.

To investigate the effectiveness of the proposed algorithms, an example of a linear system with integer inputs is adopted from the field of power electronics. More specifically, a variable speed drive system is considered that consists of a three-level neutral point clamped (NPC) inverter driving a medium-voltage (MV) induction machine (IM).

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

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

II. Optimal Control Problem

A. Mathematical Model of the System

Consider a linear system with state vector $\boldsymbol{x} \in \mathbb{R}^{n_x}$, output vector $\boldsymbol{y} \in \mathbb{R}^{n_y}$, and integer-valued input vector $\boldsymbol{u} \in \boldsymbol{\mathcal{U}}$, with $\boldsymbol{\mathcal{U}} = \mathcal{U}^{n_u}$ and $\mathcal{U} \subset \mathbb{Z}$. In the discrete-time domain, the system is described by

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

$$\boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k)\,, \tag{1b}$$

where 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 assumed to be time invariant. Finally, $k \in \mathbb{N}$ denotes the time step.

B. Model Predictive Control With Output Reference Tracking

Consider MPC with output regulation and a penalty on (switching) changes of the manipulated variable. Let $J : \mathbb{R}^{n_y} \times \boldsymbol{\mathcal{U}} \to \mathbb{R}^+$ denote the objective function

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

over a finite prediction horizon of N time steps. The output regulation error is defined as $\boldsymbol{y}_{\text{err}}(k) = \boldsymbol{y}_{\text{ref}}(k) - \boldsymbol{y}(k)$, with $\boldsymbol{y}_{\text{ref}}$ being the output reference. Changes in the manipulated variable are defined as $\Delta\boldsymbol{u}(k) = \boldsymbol{u}(k) - \boldsymbol{u}(k-1)$. The weighting matrices are defined as $\boldsymbol{Q} \in \mathbb{R}^{n_y \times n_y}, \boldsymbol{Q} \succeq 0$ and $\boldsymbol{R} \in \mathbb{R}^{n_u \times n_u}, \boldsymbol{R} \succ 0$. These set the trade-off between the tracking accuracy and the control effort.

Considering as optimization variable the integer-valued sequence of control actions over the horizon N, i.e., $\boldsymbol{U}(k) = [\boldsymbol{u}^T(k) \ \dots \ \boldsymbol{u}^T(k+N-1)]^T \in \mathbb{U} = \boldsymbol{\mathcal{U}}^N \subset \mathbb{Z}^n$ with $n = N \cdot n_u$, the following optimization problem is formulated and solved at time step k

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

In a subsequent step, the objective function (2) is written in vector form as

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

where 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 output sequence and the corresponding output reference sequence over the horizon, respectively. Moreover, the block diagonal matrices $\widetilde{\boldsymbol{Q}}$ and $\widetilde{\boldsymbol{R}}$ are defined as $\widetilde{\boldsymbol{Q}} = \oplus_{i=1}^{N} \boldsymbol{Q}$ and $\widetilde{\boldsymbol{R}} = \oplus_{i=1}^{N} \boldsymbol{R}$, while the matrices $\boldsymbol{\Gamma}$, $\boldsymbol{\Upsilon}$, $\boldsymbol{S}$ and $\boldsymbol{\Xi}$ are provided in the appendix.

Dropping the constraint that $\boldsymbol{U}(k)$ is integer-valued, i.e., allowing $\boldsymbol{U}(k) \in \mathbb{R}^n$, the unconstrained solution $\boldsymbol{U}_{\text{unc}}(k) \in \mathbb{R}^n$ of the relaxed version of the optimization problem (3) is equal to

$$\boldsymbol{U}_{\text{unc}}(k) = -\boldsymbol{W}^{-1}\boldsymbol{\Lambda}(k)\,, \tag{5}$$

where the vector $\boldsymbol{\Lambda}(k)$ is given in the appendix. After some algebraic manipulations (see [13] for more details), function (4) becomes

$$J(k) = \big(\boldsymbol{U}(k) - \boldsymbol{U}_{\text{unc}}(k)\big)^T \boldsymbol{W} \big(\boldsymbol{U}(k) - \boldsymbol{U}_{\text{unc}}(k)\big) + \text{const}(k)\,, \tag{6}$$

with $\boldsymbol{W} \in \mathbb{R}^{n \times n}$, $\boldsymbol{W} = \boldsymbol{\Upsilon}^T \, \widetilde{\boldsymbol{Q}} \, \boldsymbol{\Upsilon} + \boldsymbol{S}^T \, \widetilde{\boldsymbol{R}} \, \boldsymbol{S} \succ 0$. Considering that the constant term in (6) does not affect the result of the optimization problem, and after factoring $\boldsymbol{W}$ with the Cholesky factorization, i.e., $\boldsymbol{W} = \boldsymbol{H}^T\boldsymbol{H}$, with $\boldsymbol{H} \in \mathbb{R}^{n \times n}$ being a nonsingular, upper triangular matrix, the objective function takes its final form [15]

$$J(k) = ||\bar{\boldsymbol{U}}_{\text{unc}}(k) - \boldsymbol{H}\boldsymbol{U}(k)||^2_2\,, \tag{7}$$

with $\bar{\boldsymbol{U}}_{\text{unc}}(k) = \boldsymbol{H}\boldsymbol{U}_{\text{unc}}(k)$. With this, the optimization problem (3) can be rewritten as

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

which is an ILS problem, with $\boldsymbol{H}$ being the lattice generator matrix, which generates the space wherein the integer solution $\boldsymbol{U}^*$ lies.

To reduce the time required to solve the ILS problem (8), it is beneficial to add a preprocessing stage. This is because the lattice $\mathcal{L}(\boldsymbol{H}) = \{\sum_{i=1}^{n} \kappa_i \boldsymbol{h}_i \mid \kappa_i \in \mathbb{Z}\}$ generated by the columns of $\boldsymbol{H}$ may need reshaping to transform a potentially ill-conditioned problem into a well-conditioned one. Specifically, the goal of the preprocessing stage is to derive an upper triangular matrix $\widetilde{\boldsymbol{H}}$ with two desirable properties. First, the basis vectors of the new lattice generated by $\widetilde{\boldsymbol{H}}$ (i.e., the columns of $\widetilde{\boldsymbol{H}}$) should be close to orthogonal. Second, the length of the basis vectors (i.e., the Euclidean norm of the columns of $\widetilde{\boldsymbol{H}}$) should not be arbitrarily large so that the search space is bounded. To meet these criteria, the Lenstra-Lenstra-Lovász (LLL) lattice basis reduction algorithm [16] is employed. The resulting ILS problem takes the form [15]

$$\underset{\widetilde{\boldsymbol{U}}(k) \in \mathbb{U}}{\text{minimize}} \quad || \, \widetilde{\boldsymbol{U}}_{\text{unc}}(k) - \widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{U}}(k)||^2_2 \tag{9}$$

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

III. Solving the ILS Problem

Hereafter, three approaches are presented to solve (9). The first approach yields the optimal solution, while the second and third approach occasionally yield suboptimal solutions.

A. Optimal Solution

To solve (9) in a computationally efficient manner, a sphere decoding algorithm can be employed [11]. The sphere decoder is a depth-first search algorithm that considers only the nodes (i.e., integer points of the lattice) that lie within a hypersphere (n-dimensional sphere) of radius ρ centered at the unconstrained solution. The optimal solution is found by exploring all branches of the search tree until pruning occurs or when the bottom level is reached, at which backtracking occurs to explore unvisited nodes at higher levels. Thanks to its smart branch-and-bound mechanism, suboptimal branches of the search tree are pruned early on during the search process, whilst the integer-valued solution $\boldsymbol{U}^*$ is guaranteed to be always found.

Since the number of the nodes within the hypersphere depends on its volume, the choice of the initial radius is central. As implied, as small a radius as possible is preferred to minimize the number of nodes contained in the sphere, while ensuring that the sphere is nonempty. A trade-off solution is to compute the initial radius based on the Babai estimate [17], [18], which is found by rounding the unconstrained solution to the closest *feasible* integer vector, i.e.,

$$\boldsymbol{U}_{\text{bab}}(k) = \lfloor \boldsymbol{H}^{-1}\bar{\boldsymbol{U}}_{\text{unc}}(k) \rceil = \lfloor \boldsymbol{U}_{\text{unc}}(k) \rceil \in \mathbb{U}\,. \tag{10}$$

This choice appears to be effective since the success probability of the Babai estimate, i.e., the probability that the Babai estimate $\boldsymbol{U}_{\text{bab}}(k)$ is equal to the optimal solution $\boldsymbol{U}^*(k)$, is high when the ILS problem is well-conditioned [19]. Based on the above, the initial radius of the sphere is set to

$$\rho_1(k) = || \, \widetilde{\boldsymbol{U}}_{\text{unc}}(k) - \widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{U}}_{\text{bab}}(k)||_2 \, , \tag{11}$$

where $\widetilde{\boldsymbol{U}}_{\text{bab}}(k) = \boldsymbol{M}^{-1}\boldsymbol{U}_{\text{bab}}(k)$.

Alternatively, the initial radius can be computed by exploiting the receding horizon principle of MPC. Shifting the previously computed solution $\boldsymbol{U}^*(k-1)$ by one time step—as the receding horizon policy dictates—and repeating the last control action, yields the sequence of control actions [13]

$$\boldsymbol{U}_{\text{ed}}(k) = \boldsymbol{P}\boldsymbol{U}^*(k-1) \in \mathbb{U} \, , \tag{12}$$

where $\boldsymbol{P}$ is given in [13]. The resulting initial radius is then

$$\rho_2(k) = || \, \widetilde{\boldsymbol{U}}_{\text{unc}}(k) - \widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{U}}_{\text{ed}}(k)||_2 \, , \tag{13}$$

where $\widetilde{\boldsymbol{U}}_{\text{ed}}(k) = \boldsymbol{M}^{-1}\boldsymbol{U}_{\text{ed}}(k)$. The minimum of ρ_1 and ρ_2 is chosen as the initial radius ρ, ensuring a tight yet nonempty initial sphere.

The solution process starts by identifying the nodes that are within the sphere. To reduce the operations, the search tree is generated in a bottom-to-top manner. Since the lattice generator matrix is upper triangular, the nodes of higher dimensions are located at the top levels of the search tree and are thus explored first, while the one-dimensional nodes are located at the bottom and are visited later. When a branch has been fully explored, i.e., a full-dimensional sequence $\boldsymbol{U}$ has been assembled, the condition [12]

$$\rho^2(k) \geq \sum_{i=1}^{n} \left(\tilde{u}_{\text{unc}_i} - \sum_{j=1}^{n} \tilde{h}_{i,j}\tilde{u}_j \right)^2 \tag{14}$$

holds. If $\rho^2(k)$ exceeds the right-hand side of (14), the sphere can be tightened by updating its radius.

The algorithm of the sphere decoder is summarized in Algorithm 1. The initial values of the arguments are $\widetilde{\boldsymbol{U}} \leftarrow [\,]$, i.e., the empty vector, $d \leftarrow 0$, $i \leftarrow n$, $\rho \leftarrow \min\{\rho_1(k), \rho_2(k)\}$, and $\widetilde{\boldsymbol{U}}_{\text{unc}} \leftarrow \widetilde{\boldsymbol{H}}\,\boldsymbol{M}^{-1}\boldsymbol{U}_{\text{unc}}(k)$.

Algorithm 1 Sphere Decoder

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

B. Suboptimal Solution Based on Initial Guess

To reduce the complexity, but by sacrificing optimality, the initial guess $\boldsymbol{U}_{\text{init}}(k)$ of the integer solution can be applied to the plant without entering the optimization phase. This solution is equal to the Babai estimate (10) or the educated guess (12), depending on the corresponding radius. Specifically, the applied control sequence $\boldsymbol{U}_{\text{appl}}(k)$ is chosen as

$$\boldsymbol{U}_{\text{appl}}(k) = \boldsymbol{U}_{\text{init}}(k) = \begin{cases} \boldsymbol{U}_{\text{bab}}(k) & \text{if } \rho_1(k) \leq \rho_2(k) \\ \boldsymbol{U}_{\text{ed}}(k) & \text{if } \rho_1(k) > \rho_2(k) \end{cases} . \tag{15}$$

Since the LLL reduced lattice is nearly orthogonal, $\boldsymbol{U}_{\text{init}}(k)$ is, in general, close to the optimal solution, matching it with a high probability, as can be seen in Section V.

C. Suboptimal Solution With Stopping Criterion

In a real-time implementation, a feasible solution must be found within a given time, usually the sampling interval. To ensure this, a stopping criterion is added to the sphere decoder by imposing an upper bound on the number of flops N_t performed in real time, which directly relates to the available computational power.

This search algorithm allows to optimally operate the system when n is small (i.e., for short horizons), and to occasionally implement suboptimal solutions as the horizon increases, so that to keep the computational complexity bounded. To do so, the optimal solution $\boldsymbol{U}^*(k)$ is applied to the plant as long as the sphere decoder manages to find it *without* violating the flop count upper bound. If, on the other hand, the maximum allowable flops N_t^u have been performed and the sphere decoder has not terminated yet, then the last fully computed sequence of control actions $\boldsymbol{U}_{\text{sub}}(k)$ is implemented[1]. Note that, $J_{\boldsymbol{U}^*} \leq J_{\boldsymbol{U}_{\text{sub}}} \leq J_{\boldsymbol{U}_{\text{init}}}$, where $J_{\boldsymbol{U}^*}$, $J_{\boldsymbol{U}_{\text{sub}}}$, and $J_{\boldsymbol{U}_{\text{init}}}$ are the costs computed when applying $\boldsymbol{U}^*$, $\boldsymbol{U}_{\text{sub}}$, and $\boldsymbol{U}_{\text{init}}$, respectively. Hence, the applied control action $\boldsymbol{U}_{\text{appl}}(k)$ is defined as

$$\boldsymbol{U}_{\text{appl}}(k) = \begin{cases} \boldsymbol{U}^*(k) & \text{if } N_t \leq N_t^u \\ \boldsymbol{U}_{\text{sub}}(k) & \text{if } N_t > N_t^u \end{cases} . \tag{16}$$

IV. Computational Complexity Analysis

In this section, an analysis is presented of the computational complexity of the three algorithms proposed in Section III. The analysis focuses only on the flops performed in real time while solving (9), whereas those required in the formulation and preprocessing stage of the ILS problem are neglected (see [15] for more details).

First, the flops required to compute the unconstrained solution $\boldsymbol{U}_{\text{unc}}$ and the initial radius ρ are counted. Assuming that the matrix $\boldsymbol{W}$ (see (6)) is time-invariant and in general dense, then for finding the unconstrained solution requires

[1] If a control sequence has not been assembled within the given time, then the initial guess is implemented, i.e., $\boldsymbol{U}_{\text{sub}}(k) = \boldsymbol{U}_{\text{init}}(k)$.

roughly[2] $n(2n-1)$ flops [20] ($n(n-1)$ additions and n^2 multiplications). Moreover, n^2+3n-1 flops are required for the computation of the initial (squared) radius ρ^2, considering that $\widetilde{\boldsymbol{H}}$ is upper triangular. In particular, $n(n-1)/2+n-1$ additions, $n(n+1)/2+n$ multiplications, and n subtractions are performed, see e.g. (11). However, since the radius is computed based on two different approaches ((11) and (13)), the required flops are doubled[3].

Due to the fact that the ILS problem is NP-hard [18], [21], a tight upper bound on the computational complexity of the algorithm employed to solve it cannot be given. Instead, in order to indicate the complexity of the sphere decoder, we focus on one call of the recursive Algorithm 1. As can be seen, the operations performed by the sphere decoder are a function of the dimension of the node visited. Thus, for the computation of the Euclidean distance between an m-dimensional node, with $m=1,\ldots,n$, and the unconstrained solution (see line 4 in Algorithm 1), $n-m+1$ additions[4] ($n-m$ for the computation of $\hat{\widetilde{\boldsymbol{H}}}_{(m,m:n)}\,\widetilde{\boldsymbol{U}}_{m:n}$ and one for the addition $||*||_2^2+d^2$), one subtraction, and $n-m+2$ multiplications ($n-m+1$ for the computation of $\hat{\widetilde{\boldsymbol{H}}}_{(m,m:n)}\hat{\widetilde{\boldsymbol{U}}}_{m:n}$ and one for squaring the Euclidean norm $||*||_2^2$) are performed. Moreover, considering that a parent node has $|\mathcal{U}|$ children nodes, where $|\mathcal{U}|$ denotes the cardinality of $\mathcal{U}$, then for each m-dimensional child node explored, the remaining $|\mathcal{U}|-1$ children nodes of the same parent (sibling nodes) have to be evaluated to determine whether they lie inside the hypersphere. Finally, it should be mentioned that divisions are not performed.

Based on the above flop count analysis, the total number of additions N_a, subtractions N_s and multiplications N_m performed in real time is

$$\begin{aligned} N_a &= 2(n^2-1)+|\mathcal{U}|\left(\mu-1+\sum_{\nu=1}^{\mu}\big(n-m(\nu)\big)\right),\\ N_s &= 2n+|\mathcal{U}|\mu\,,\\ N_m &= 2n(n+3/2)+|\mathcal{U}|\left(2\mu+\sum_{\nu=1}^{\mu}\big(n-m(\nu)\big)\right), \end{aligned} \tag{17}$$

where μ denotes the number of nodes explored by the sphere decoder.

As a result, $N_t=N_a+N_s+N_m$ flops must be performed at each time step when executing the MPC scheme. Hence, the upper bound on N_t corresponds to the worst-case scenario, i.e., the case where (a) the maximum number of nodes $\max(\mu)$ is explored by the sphere decoder, and (b) the input vector $\boldsymbol{U}\in\mathcal{B}$, with $\mathcal{B}=\mathbb{Z}^n\setminus\{-1,0,1\}$. The latter can be justified if we consider that if $\boldsymbol{U}\notin\mathcal{B}$, then $n-m+1$ multiplications performed at the mth level by the sphere decoder (see line 4 in Algorithm 1) will result in either the multiplicand itself, with the same or reversed sign, or zero. This means that only one multiplication will be essentially performed at each node, regardless of its dimension, whereas multiplications with zero would result in a decrease in the number of additions as well[5].

[2]Vector $\boldsymbol{\Lambda}(k)$ has also to be computed in real time since it is time variant, see the appendix. However, the number of operations required is $O(n^2)$.

[3]This holds only when $u_{\text{bab}_i}\neq u_{\text{ed}_i}\ \forall\, i=1,\ldots,n$.

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

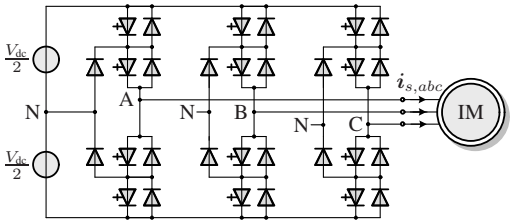


Fig. 1: Three-level three-phase neutral point clamped (NPC) voltage source inverter driving an induction machine with a fixed neutral point potential.

Taking the above into account, the three algorithms proposed in Section III are of different complexity. The running time of the sphere decoder, i.e., the search algorithm that yields the optimal solution (Section III-A), cannot be bounded by a polynomial since it depends not only on the size of the input n, i.e., the prediction horizon and the size of the control input, but also on the number of nodes visited, i.e., the number of the recursive calls. On the other hand, the algorithm that applies the initial guess (Section III-B) has quadratic running time $O(n^2)$, which is much more efficient than the sphere decoder. Finally, the complexity of the suboptimal algorithm with the stopping criterion (Section III-C) is bounded by a constant—the flop count upper bound—as n becomes large. This algorithm appears to be the most efficient as n increases, but for smaller n the initial guess algorithm is faster, as also shown in Section V.

V. Case Study

The search algorithms introduced in Section III are tested on an industrial case study, namely a medium-voltage drive system consisting of a three-level NPC voltage source inverter and a squirrel cage induction machine, see Fig. 1. The inverter is supplied by a constant dc-link voltage $V_{\text{dc}}=5.2\,\text{kV}$ and has a fixed neutral point potential. The machine has $3.3\,\text{kV}$ rated voltage, $356\,\text{A}$ rated current, $2\,\text{MVA}$ rated power, $50\,\text{Hz}$ nominal frequency, and 0.25 per unit (p.u.) total leakage inductance. The proposed MPC strategy directly controls the stator currents of the machine, thus an intermediate modulation stage is not necessary. The sampling interval used is $T_s=25\,\mu\text{s}$.

A. Internal Dynamic Model of the System

As shown in Fig. 1, the output phase voltages can assume the three discrete voltage levels $-\frac{V_{\text{dc}}}{2}$, 0, and $\frac{V_{\text{dc}}}{2}$, depending on the position of the switches in the respective phase leg. These positions, in turn, can be described by the integer variables $u_a,u_b,u_c\in\mathcal{U}=\{-1,0,1\}$, which are the manipulated variables. We define $\boldsymbol{u}=[u_a\ u_b\ u_c]^T\in\boldsymbol{\mathcal{U}}$.

Selecting the stator current $\boldsymbol{i}_{s,\alpha\beta}$ and the rotor flux $\boldsymbol{\psi}_{r,\alpha\beta}$ in the $\alpha\beta$ plane[6] as the state variables, i.e.,

[5]Note that the same argument holds for the computation of ρ.

[6]To ease the calculations the three-phase system (abc) variables are transformed to the stationary orthogonal $\alpha\beta$ system via the appropriate $\boldsymbol{K}$ matrix, i.e., $\boldsymbol{\xi}_{\alpha\beta}=\boldsymbol{K}\boldsymbol{\xi}_{abc}$, where $\boldsymbol{\xi}_{abc}=[\boldsymbol{\xi}_a\ \boldsymbol{\xi}_b\ \boldsymbol{\xi}_c]^T$ is any variable in the abc plane, and $\boldsymbol{\xi}_{\alpha\beta}=[\boldsymbol{\xi}_\alpha\ \boldsymbol{\xi}_\beta]^T$ the resulting variable in the $\alpha\beta$ plane.

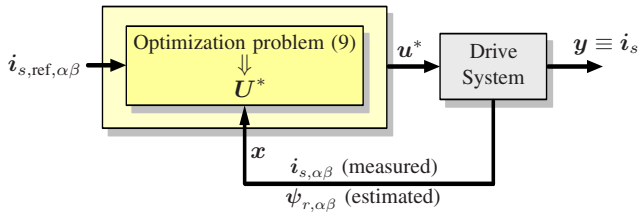


Fig. 2: Model predictive current control with reference tracking for the three-phase three-level NPC inverter with an induction machine.

$\boldsymbol{x} = [i_{s\alpha}\ i_{s\beta}\ \psi_{r\alpha}\ \psi_{r\beta}]^T \in \mathbb{R}^4$, and the stator current as the output of the system, $\boldsymbol{y} = [i_{s\alpha}\ i_{s\beta}]^T \in \mathbb{R}^2$, a discrete-time state-space model of the drive system in the form (1) results, with $n_x = 4$, $n_y = 2$ and $n_u = 3$. The matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ are calculated using exact Euler discretization, i.e., they are of the form $\boldsymbol{A} = \mathbf{e}^{\boldsymbol{F}T_s}$, $\boldsymbol{B} = -\boldsymbol{F}^{-1}(\boldsymbol{I} - \boldsymbol{A})\boldsymbol{G}$ and $\boldsymbol{C} = \boldsymbol{E}$, where $\boldsymbol{F}$, $\boldsymbol{G}$, and $\boldsymbol{E}$ are the matrices of the continuous state-space representation of the drive and can be found in [15], and $\mathbf{e}$ the matrix exponential.

B. Direct Model Predictive Current Control

The main control objectives are the tracking of the stator current reference $\boldsymbol{i}_{s,\mathrm{ref},\alpha\beta}$ and the minimization of the switching (i.e., control) effort, in order to minimize the switching losses. To do so, the switches of the inverter are *directly* manipulated in such a way that the transitions from one state to another are minimized, while eliminating the current error. The block diagram of the controller is summarized in Fig. 2

To fulfill the above-mentioned control goals, the objective function consists of two terms: a current error term and a control effort term. As a result, the objective function J to be minimized is of the form (2), where $\boldsymbol{y}_{\mathrm{err}} = \boldsymbol{i}_{\mathrm{err},\alpha\beta} = \boldsymbol{i}_{s,\mathrm{ref},\alpha\beta} - \boldsymbol{i}_{s,\alpha\beta}$ and $\boldsymbol{Q} = \boldsymbol{I}$ (since the $\alpha-$ and $\beta-$components of the stator current are of the same magnitude) and $\boldsymbol{R} = \lambda_u \boldsymbol{I}$, with $\lambda_u \in \mathbb{R}^+$.

C. Performance Evaluation

Simulation results are presented in this section that highlight the potential benefits of the proposed algorithms. First, the steady-state performance of the drive system is investigated when applying the optimal solution to it. For the horizon $N = 10$, the weighting factor is set to $\lambda_u = 0.1$. This results in a switching frequency—corresponding to the control effort—of approximately 300 Hz. In Fig. 3(a), the normalized three-phase stator current waveforms and their references are illustrated over one fundamental period. The resulting current spectrum shown in Fig. 3(b) highlights the tracking performance of the controller. The latter can be quantified using the total harmonic distortion (THD) of the current. The current THD being 4.95%, we conclude that the tracking performance of the controller is good, considering the low switching frequency. Finally, Fig. 3(c) depicts the three-phase switch positions, the manipulated variable, over one fundamental period.

The current THD is depicted in Fig. 4 as a function of the prediction horizon. The figure shows the THD resulting from the three algorithms employed to solve the ILS problem underlying direct MPC. The weighting factor λ_u is tuned such that the switching frequency is about 300 Hz, regardless of the prediction horizon. The system performance does not deteriorate significantly when the suboptimal algorithms are used. This results from the fact that in the vast majority of cases the applied control input is the optimal one. This can be seen in Table I, where it is shown the percentage of times the optimal solution is implemented when using the two aforementioned strategies.

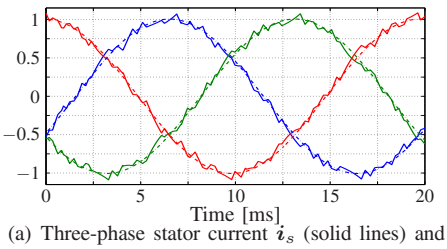
TABLE I: The percentage of times the optimal solution is applied when using the suboptimal algorithm based either on (a) the initial guess (Section III-B), or (b) the stopping criterion (Section III-C), depending on the length of the prediction horizon N.

Prediction horizon N	$\boldsymbol{U}_{\mathrm{appl}} = \boldsymbol{U}^*$ %	
	Initial Guess	Stopping Criterion
1	99.4	100
2	99.2	100
3	98.9	100
4	98.5	100
5	97.9	100
7	97.0	100
10	95.7	99.1

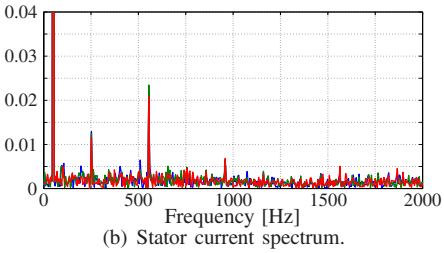
Finally, the computational complexity—in terms of the flop count—of the three algorithms presented in Section III is investigated (Fig. 5). The flops required by the exhaustive enumeration are also presented. The latter algorithm is used to define the flop count upper bound in the search algorithm with the stopping criterion. Specifically, the upper bound is set equal to the flops performed by a two-step MPC algorithm with exhaustive enumeration, assuming that this algorithm can be executed in real time by a microprocessor or a field-programmable gate array (FPGA). Thus, the maximum allowable number of flops performed by the sphere decoder is $4{,}978 - 3n^2 - 4n + 1$, based on the analysis given in Section IV.

VI. Conclusions

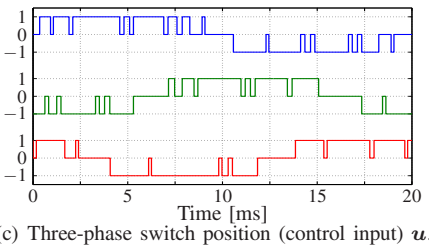
The optimization problem underlying model predictive control (MPC) of linear systems with integer inputs can be formulated as an integer least-squares (ILS) problem. This paper presented three algorithms that solve the ILS problem in a computationally efficient manner. The computational complexity of these strategies—as measured by the flops required in real time—varies depending on the optimality result. When the solution found by the optimal search algorithm, named sphere decoder, is always implemented, thus, the most desirable performance of the plant is achieved, the flop count cannot be bounded by a polynomial expression. If, on the other hand, we want to keep the flop count low, by applying the initial guess in quadratic time, then the performance of the system is slightly deteriorated since, in some cases, it is a suboptimal solution. Finally, a suboptimal algorithm with a stopping criterion has a computational complexity which is bounded by a constant, i.e., a scalar that corresponds to the maximum allowable real-time flops, whereas the plant performance is kept close to the optimal. An industrial three-level inverter drive system was considered as an illustrative example to highlight the potential benefits of each algorithm.



(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. 3: 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%.

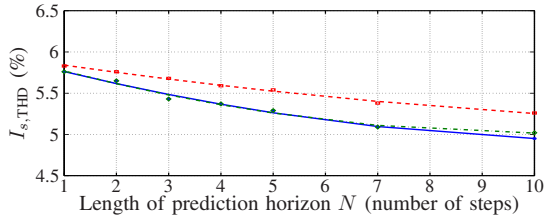


Fig. 4: Stator current THD as a function of the prediction horizon N for the three solution algorithms. The data points relate to individual simulations, which were approximated using polynomial functions of second order. Specifically, the (blue) stars and solid line refers to the optimal solution, the (red) squares and dashed line to the suboptimal solution based on the initial guess, while the (green) rhombi and dash-dotted line corresponds to the suboptimal algorithm with the stopping criterion.

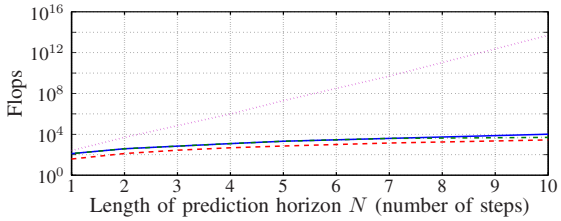


Fig. 5: Flop count as a function of the prediction horizon N as resulting from the three different search algorithms and the exhaustive enumeration. The solid (blue) line refers to the algorithm that yields the optimal solution, the dashed (red) line to the one that implements the initial guess, the dash-dotted (green) line corresponds to the suboptimal algorithm with the stopping criterion, and the dotted (magenta) line to the exhaustive enumeration.

Appendix

The matrices $\boldsymbol{\Upsilon}$, $\boldsymbol{\Gamma}$, $\boldsymbol{S}$, and $\boldsymbol{\Xi}$ in (4) are defined as

$$\boldsymbol{\Upsilon} = \begin{bmatrix} \boldsymbol{CB} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{CAB} & \boldsymbol{CB} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & & \vdots \\ \boldsymbol{CA}^{N-1}\boldsymbol{B} & \boldsymbol{CA}^{N-2}\boldsymbol{B} & \cdots & \boldsymbol{CB} \end{bmatrix},$$

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

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

Vector $\boldsymbol{\Lambda}(k)$ in (5) is

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

References

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

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

[3] M. Schmitt, R. Vujanic, J. Warrington, and M. Morari, "An approach for model predictive control of mixed integer-input linear systems based on convex relaxations," in *Proc. IEEE Conf. Decis. Control*, Florence, Italy, Dec. 2013, pp. 6428–6433.

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

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

[6] D. Bertsimas and Y. Ye, "Semidefinite relaxations, multivariate normal distributions, and order statistic," in *Hdbk of Combinat. Optim.*, D.-Z. Du and P. M. Pardalos, Eds. New York, NY: Kluwer Academic, 1998, vol. 3, pp. 1–19.

[7] G. C. Goodwin and D. Quevedo, "Finite alphabet control and estimation," *Int. J. of Control, Autom. & Syst.*, vol. 42, no. 6, pp. 412–430, Dec. 2003.

[8] A. Billionnet and S. Elloumi, "Using a mixed integer quadratic programming solver for the unconstrained quadratic $0-1$ problem," *Math. Program.*, vol. 109, no. 1, pp. 55–68, Jan. 2007.

[9] D. Axehill, L. Vandenberghe, and A. Hansson, "Convex relaxations for mixed integer predictive control," *Automatica*, vol. 46, no. 9, pp. 1540–1545, Sep. 2010.

[10] 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.

[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] 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.

[14] ——, "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.

[15] 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.

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

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

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

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

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

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