Reformulation of the long-horizon direct model predictive control problem to reduce the computational effort

Reformulation of the Long-Horizon Direct Model Predictive Control Problem to Reduce the Computational Effort

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

***Abstract*—For direct model predictive control schemes with current reference tracking, the underlying integer least-squares (ILS) problem is reformulated to reduce the computational complexity of the solution stage. This is achieved by exploiting the geometry of the ILS problem and by reducing the computations needed for its formulation and solution. A lattice reduction and a sphere decoding algorithm are implemented. 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.**

I. Introduction

Model predictive control (MPC) [1] is a control strategy that has recently gained popularity in the field of power electronics due to its numerous advantages [2]. MPC is suitable for nonlinear multiple-input multiple-output (MIMO) plants with complex dynamics. Furthermore, constraints can be explicitly or implicitly imposed and met, by formulating the control problem as a constrained optimization problem. Thanks to the so-called receding horizon policy, disturbances are effectively rejected and a high degree of robustness to model uncertainties is generally achieved.

However, MPC schemes are often computationally demanding, especially when implemented as a direct controller. Direct MPC schemes manipulate the inverter switch positions without the use of a modulator, giving rise to a switched linear or nonlinear system model. These MPC problems are often solved by enumerating all candidate solutions, implying that all possible solutions are exhaustively evaluated to determine the optimal one. Since long prediction horizons are often required to ensure stability and good closed-loop performance [3], the underlying optimization problem may become computationally intractable.

As a result, the horizon is typically kept as short as possible. In power electronics, a one-step horizon is often used, especially when reference tracking of the variables of interest is considered for power electronic systems of first order [2]. Despite a few strategies to make the implementation of MPC algorithms with long prediction horizons feasible in real time [4], computational issues prevail, which need to be addressed to enable the widespread use of direct MPC algorithms in power electronics.

Recently, an optimization technique was proposed in [5] and evaluated in [6] that solves the direct MPC problem with only a modest increase in the computational burden when enlarging the prediction horizon. The underlying optimization problem is formulated as an integer least-squares (ILS) problem and the branch-and-bound technique of sphere decoding [7] is adopted to compute the optimal solution (the optimal switching sequence). It is expected that the complexity of the problem does not significantly increase when stepping up the number of converter voltage levels, making this method particularly attractive for multilevel topologies.

By reformulating the ILS problem through a preprocessing stage, this paper proposes a modified version of the algorithm introduced in [5]. A lattice reduction algorithm reduces the size of the n-dimensional solution 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 [5] is refined. With these improvements, the computations to be performed in real time are reduced, while still obtaining the optimal solution. As a case study, a variable speed drive system is considered, which consists of a three-level neutral point clamped (NPC) voltage source inverter driving a medium-voltage (MV) induction machine (IM). For long horizons, such as ten steps, the number of nodes to be evaluated is reduced by up to 45%, highlighting the efficacy of the proposed algorithm.

II. Formulation of the Optimal Control Problem

For reasons of simplicity, in this work the optimal control problem to be solved in the framework of MPC considers the direct current control problem in an IM drive. A three-level NPC voltage source inverter is used with a fixed neutral point potential and a constant dc-link voltage V_{dc}, see Fig. 1. It is important to note that the formulation of the problem and the analysis that follows can be easily extended to other converter topologies and control tasks.

A. Mathematical Model of the System

When deriving the prediction model of the plant, it is common practice to transform the variables from the three-

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

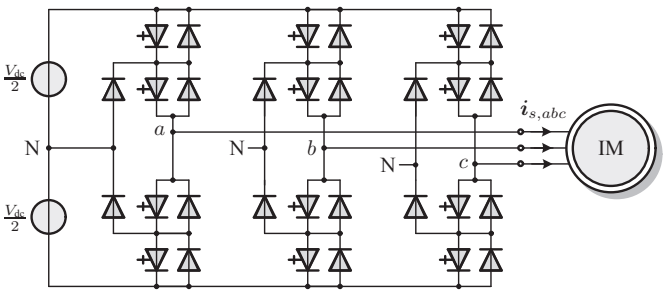


Fig. 1: Three-level three-phase neutral point clamped (NPC) voltage source inverter driving an induction motor (IM) with a fixed neutral point potential.

phase (abc) to the stationary orthogonal $\alpha\beta$ system. For this, we define $\boldsymbol{\xi}_{\alpha\beta} = \boldsymbol{K}\boldsymbol{\xi}_{abc}$, where $\boldsymbol{\xi}_{abc} = [\xi_a \ \xi_b \ \xi_c]^T$ is a vector in the abc plane, and $\boldsymbol{\xi}_{\alpha\beta} = [\xi_\alpha \ \xi_\beta]^T$ is the resulting vector in the $\alpha\beta$ plane, with the transformation matrix

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

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 phase switches. The switch positions in the phase legs can be described by the integer variables $u_a, u_b, u_c \in \mathcal{U} = \{-1, 0, 1\}$. Hence, the output voltage of the inverter is given by[1]

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

where $\boldsymbol{u} = [u_a \ u_b \ u_c]^T$.

To derive the state-space model of the squirrel-cage IM in the $\alpha\beta$ plane, the stator current $\boldsymbol{i}_{s,\alpha\beta}$ and the rotor flux $\boldsymbol{\psi}_{r,\alpha\beta}$ are chosen as state variables. It should be mentioned that the dynamic of the rotor angular speed ω_r is neglected, since the speed is considered to be a time-varying parameter. The model input is the stator voltage $\boldsymbol{v}_{s,\alpha\beta}$, which is equal to the inverter output voltage, as given by (1). The model parameters are the stator R_s and rotor R_r resistances, the stator X_{ls}, rotor X_{lr} and mutual X_m reactances, the inertia J, and the mechanical load torque T_ℓ.

Considering the above, the continuous-time state equations are[2] [8]

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

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

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

with $\Phi = X_s X_r - X_m^2$, $X_s = X_{ls} + X_m$ and $X_r = X_{lr} + X_m$. Furthermore, in (2) $\tau_s = X_r\Phi/(R_s X_r^2 + R_r X_m^2)$ and $\tau_r = X_r/R_r$ are the stator and rotor time constants, respectively, $\boldsymbol{I}$ is the identity matrix of appropriate dimension (here two by two), whereas T_e is the electromagnetic torque.

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

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

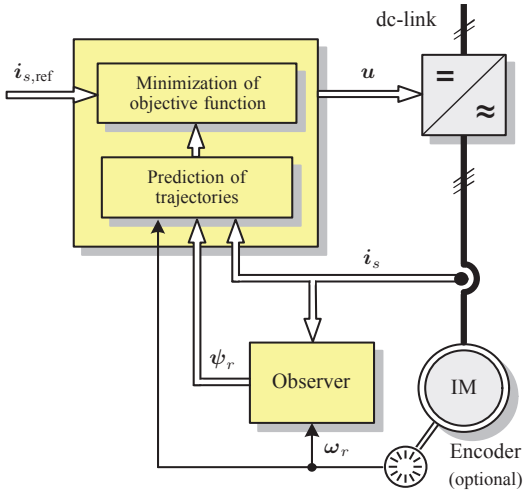


Fig. 2: Model predictive current control with reference tracking for the three-phase three-level NPC inverter with an IM.

Given the model of the drive, as described by (1) and (2), the state-space model in the continuous-time domain is

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

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

In (3) $\boldsymbol{x} = [i_{s\alpha} \ i_{s\beta} \ \psi_{r\alpha} \ \psi_{r\beta}]^T$ is the state vector, encompassing the stator current and the rotor flux in the $\alpha\beta$ plane. The output of the system is the stator current, i.e. $\boldsymbol{y} = \boldsymbol{i}_{s,\alpha\beta}$. The switch position $\boldsymbol{u} = [u_a \ u_b \ u_c]^T$ constitutes the input vector provided by the controller. Finally, the continuous-time matrices $\boldsymbol{D}$, $\boldsymbol{E}$, and $\boldsymbol{F}$ can be calculated using elementary linear algebra.

The state-space model of the drive can be transformed from the continuous-time domain to the discrete-time domain using exact Euler discretization. The discrete-time representation is

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

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

The matrices $\boldsymbol{A}, \boldsymbol{B}$ and $\boldsymbol{C}$ are of the form $\boldsymbol{A} = \mathrm{e}^{\boldsymbol{D}T_s}$, $\boldsymbol{B} = -\boldsymbol{D}^{-1}(\boldsymbol{I} - \boldsymbol{A})\boldsymbol{E}$ and $\boldsymbol{C} = \boldsymbol{F}$. Moreover, $\boldsymbol{I}$ is—as previously defined—the identity matrix of appropriate dimensions, e the matrix exponential, T_s the sampling interval, and $k \in \mathbb{N}$.

B. Direct Model Predictive Control with Current Reference Tracking

The block diagram of the MPC scheme with current reference tracking is shown in Fig. 2. The main control objective is the elimination of the current error, or equivalently, the accurate tracking of the reference value $\boldsymbol{i}_{s,\text{ref},\alpha\beta}$ of the stator current $\boldsymbol{i}_{s,\alpha\beta}$. To do so, the switches of the inverter are directly manipulated without the presence of a modulator. Moreover, since for MV drives switching losses are of paramount importance, the switching effort is to be kept small.

Based on the above, the objective function that penalizes at step k the evolution of the current error and the control effort over the finite prediction horizon N is

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

where $\boldsymbol{i}_{e,\alpha\beta}(\ell+1) = \boldsymbol{i}_{s,\text{ref},\alpha\beta}(\ell+1) - \boldsymbol{i}_{s,\alpha\beta}(\ell+1)$ and $\Delta \boldsymbol{u}(\ell) = \boldsymbol{u}(\ell) - \boldsymbol{u}(\ell-1)$. In (5) the weighting factor $\lambda_u \in \mathbb{R}^+$ sets the trade-off between the stator current tracking accuracy (i.e. the deviation of the current from its reference) and the switching effort (i.e. the switching frequency).

To obtain the optimal solution (i.e. the control input), the objective function (7) is minimized at time-step k, subject to the system dynamics and additional constraints (e.g. the switches are not allowed to change from $u=1$ to $u=-1$, and vice versa, directly in one step). Thus, the following optimization problem is formulated and solved in real time

$$\underset{\boldsymbol{U}(k)}{\text{minimize}} \quad J(k) \text{ (see (5))} \quad (6a)$$

$$\text{subject to} \quad \boldsymbol{U}(k) \in \mathbb{U} \quad (6b)$$

$$||\Delta \boldsymbol{u}(\ell)||_\infty \leq 1 , \ \forall \ell = k, \dots, k+N-1 . \quad (6c)$$

The optimization variable is the switching sequence $\boldsymbol{U}(k) = [\boldsymbol{u}^T(k) \dots \boldsymbol{u}^T(k+N-1)]^T \in \mathbb{R}^n$, with $n = 3N$. The set $\mathbb{U} = \boldsymbol{\mathcal{U}} \times \cdots \times \boldsymbol{\mathcal{U}}$ is the N-times Cartesian product of the set $\boldsymbol{\mathcal{U}}$, with $\boldsymbol{\mathcal{U}} = \mathcal{U} \times \mathcal{U} \times \mathcal{U}$ being the set of discrete three-phase switch positions. The second constraint in the optimization problem, see (6c), is imposed to avoid a shoot through.

C. Formulation of the ILS Problem

Following, by introducing the vectors $\boldsymbol{Y}(k)$ and $\boldsymbol{Y}_{\text{ref}}(k)$ to denote the output sequence and the corresponding output reference sequence over the horizon, respectively, i.e. $\boldsymbol{Y}(k) = [\boldsymbol{y}^T(k+1) \dots \boldsymbol{y}^T(k+N)]^T = \boldsymbol{\Gamma}\boldsymbol{x}(k) + \boldsymbol{\Upsilon}\boldsymbol{U}(k)$ and $\boldsymbol{Y}_{\text{ref}}(k) = [\boldsymbol{y}_{\text{ref}}^T(k+1) \dots \boldsymbol{y}_{\text{ref}}^T(k+N)]^T$, the objective function J can be written in vector form as

$$J = ||\boldsymbol{\Gamma}\boldsymbol{x}(k) + \boldsymbol{\Upsilon}\boldsymbol{U}(k) - \boldsymbol{Y}_{\text{ref}}||_2^2 + \lambda_u ||\boldsymbol{S}\boldsymbol{U}(k) - \boldsymbol{\Xi}\boldsymbol{u}(k-1)||_2^2 , \quad (7)$$

where the definition of matrices $\boldsymbol{\Gamma}$, $\boldsymbol{\Upsilon}$, $\boldsymbol{S}$ and $\boldsymbol{\Xi}$ can be found in the appendix. After some algebraic manipulations[3] the objective function (7) can be written as

$$J = (\boldsymbol{U}(k) - \boldsymbol{U}_{\text{unc}}(k))^T \boldsymbol{Q} (\boldsymbol{U}(k) - \boldsymbol{U}_{\text{unc}}(k)) + \text{const}(k) , \quad (8)$$

where $\boldsymbol{U}_{\text{unc}} \in \mathbb{R}^n$ is the unconstrained solution of the optimization problem (6). The matrix $\boldsymbol{Q} = \boldsymbol{\Upsilon}^T\boldsymbol{\Upsilon} + \lambda_u \boldsymbol{S}^T\boldsymbol{S}$ is by definition positive definite and can thus be factored (using Cholesky factorization) as

$$\boldsymbol{Q} = \boldsymbol{H}^T\boldsymbol{H} , \quad (9)$$

where $\boldsymbol{H} \in \mathbb{R}^{n \times n}$ is a nonsingular, upper triangular matrix. Taking (9) into account, (8) takes the form

$$J = (\boldsymbol{H}\boldsymbol{U}(k) - \bar{\boldsymbol{U}}_{\text{unc}}(k))^T (\boldsymbol{H}\boldsymbol{U}(k) - \bar{\boldsymbol{U}}_{\text{unc}}(k)) + \text{const}(k) , \quad (10)$$

[3]For a detailed derivation of (8) the reader is referred to [5].

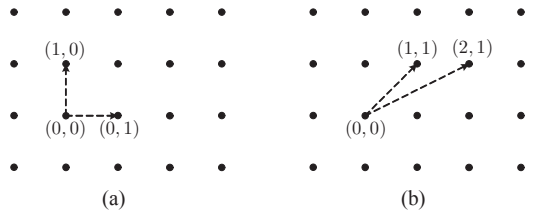


Fig. 3: A two-dimensional lattice generated by two different bases. The arrows (shown as dashed lines) correspond to the basis vectors of the lattice. In (a) the basis vectors are $\boldsymbol{h}_1 = (1, 0)$ and $\boldsymbol{h}_2 = (0, 1)$, whereas in (b) the basis vectors are $\tilde{\boldsymbol{h}}_1 = (1, 1)$ and $\tilde{\boldsymbol{h}}_2 = (2, 1)$. The two lattices are $\mathbb{Z}^2$, i.e. $\mathcal{L}(\boldsymbol{H}) = \mathcal{L}(\widetilde{\boldsymbol{H}}) = \mathbb{Z}^2$.

where $\bar{\boldsymbol{U}}_{\text{unc}} \in \mathbb{R}^n$ is

$$\bar{\boldsymbol{U}}_{\text{unc}}(k) = \boldsymbol{H}\boldsymbol{U}_{\text{unc}}(k) . \quad (11)$$

Since the constant in (10) does not affect the result of the optimization problem, the function to be minimized is

$$J = ||\bar{\boldsymbol{U}}_{\text{unc}}(k) - \boldsymbol{H}\boldsymbol{U}(k)||_2^2 . \quad (12)$$

Thus, the optimization problem (6) can be reformulated as

$$\underset{\boldsymbol{U}(k)}{\text{minimize}} \quad ||\bar{\boldsymbol{U}}_{\text{unc}}(k) - \boldsymbol{H}\boldsymbol{U}(k)||_2^2 \quad (13a)$$

$$\text{subject to} \quad \boldsymbol{U}(k) \in \mathbb{U} \quad (13b)$$

$$||\Delta \boldsymbol{u}(\ell)||_\infty \leq 1 , \ \forall \ell = k, \dots, k+N-1 . \quad (13c)$$

The reformulated optimization problem is an integer least-squares (ILS) problem. The matrix $\boldsymbol{H}$ is known as the lattice generator matrix—its columns represent n linearly independent vectors in $\mathbb{R}^n$ that generate a lattice (see Fig. 3), i.e. the set of all linear combinations

$$\mathcal{L}(\boldsymbol{H}) = \{ \sum_{i=1}^{n} w_i \boldsymbol{h}_i \mid w_i \in \mathbb{Z} \} \quad (14)$$

of the basis vectors $\boldsymbol{h}_i$.

III. Solving the ILS Problem

To facilitate the real-time implementation of the proposed MPC algorithm, the ILS problem (13) 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 (13), whereas in the second stage (optimization) the *search* for the optimal solution is performed.

A. Reduction of the ILS Problem

In a first step, the Lenstra-Lenstra-Lovász (LLL) lattice basis reduction algorithm as proposed in [9] is applied to the problem (13) to reduce the complexity of the search process. Despite the fact that $\boldsymbol{H}$ is already upper triangular, a factorization procedure is employed to transform $\boldsymbol{H}$ to the new upper triangular matrix $\widetilde{\boldsymbol{H}} \in \mathbb{R}^{n \times n}$. Thanks to the sparsity of $\boldsymbol{H}$, this procedure can be accomplished with only a few computations.

The reduction process derives the upper triangular matrix $\widetilde{\boldsymbol{H}}$ with positive diagonal entries that satisfy the following criteria:

$$|\tilde{h}_{i,j}| \leq \frac{1}{2}\tilde{h}_{i,i}, \quad i = 1, \ldots, j-1 \tag{15a}$$

$$\delta \tilde{h}^2_{j-1,j-1} \leq \tilde{h}^2_{j-1,j} + \tilde{h}^2_{j,j}, \quad j = 2, \ldots, n\,, \tag{15b}$$

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

$$\boldsymbol{M}_{i,j} = \boldsymbol{I} - \gamma \boldsymbol{e}_i \boldsymbol{e}_j^T\,, \tag{16}$$

are applied to the right-hand side of $\boldsymbol{H}$. In (16) $\boldsymbol{e}_i$ 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

$$\widetilde{\boldsymbol{H}} = \boldsymbol{H}\boldsymbol{M}_{i,j} = \boldsymbol{H} - \gamma \boldsymbol{H}\boldsymbol{e}_i \boldsymbol{e}_j^T\,. \tag{17}$$

The matrices $\boldsymbol{H}$ and $\widetilde{\boldsymbol{H}}$ are the same except for the (k,i) entries, with $k = 1, 2, \ldots, i$, which are

$$\tilde{h}_{k,j} = h_{i,j} - \gamma h_{k,i}\,, \tag{18}$$

By following a procedure similar to Gaussian elimination, i.e. by setting[4]

$$\gamma = \lfloor h_{i,j}/h_{i,i} \rceil\,, \tag{19}$$

we ensure that the entry $|\tilde{h}_{i,j}|$ is sufficiently reduced such that it meets condition (15a).

To satisfy the second condition (15b), 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 $\boldsymbol{G}_{j-1,j}$ [10], [11] is employed to maintain the upper triangular form of the resulting matrix $\widetilde{\boldsymbol{H}}$. As a result, the LLL reduced matrix $\widetilde{\boldsymbol{H}}$ can be written in its final form

$$\widetilde{\boldsymbol{H}} = \boldsymbol{G}^T_{j-1,j} \boldsymbol{H} \boldsymbol{P}_{j-1,j}\,, \tag{20}$$

where the Givens matrix is

$$\boldsymbol{G}_{j-1,j} = \begin{bmatrix} \boldsymbol{I}_{j-2} & & & \\ & c & -s & \\ & s & c & \\ & & & \boldsymbol{I}_{n-j} \end{bmatrix},$$

with

$$c^2 + s^2 = 1\,,\ c = \frac{h_{j-1,j}}{\sqrt{h^2_{j-1,j} + h^2_{j,j}}}\,,\ s = \frac{h_{j,j}}{\sqrt{h^2_{j-1,j} + h^2_{j,j}}}\,.$$

The permutation matrix in (20) is of the form

$$\boldsymbol{P}_{j-1,j} = \begin{bmatrix} \boldsymbol{I}_{j-2} & & & \\ & 0 & 1 & \\ & 1 & 0 & \\ & & & \boldsymbol{I}_{n-j} \end{bmatrix},$$

[4]With $\lfloor \boldsymbol{\xi} \rceil$ is denoted the nearest integer vector to $\boldsymbol{\xi}$, i.e. each entry of $\boldsymbol{\xi}$ is rounded to the nearest integer. Moreover, $\lceil \boldsymbol{\xi} \rceil$ denotes rounding to the nearest largest integer vector, and $\lfloor \boldsymbol{\xi} \rfloor$ rounding to the nearest smallest integer vector.

Algorithm 1 LLL Reduction

function $\widetilde{\boldsymbol{H}} = \mathrm{LLL}(\boldsymbol{H})$
  while $j \leftarrow 2 \leq n$ **do**
    if $|h_{j-1,j}| > \frac{1}{2}|h_{j-1,j-1}|$ **then**
      $\widetilde{\boldsymbol{H}} \leftarrow \boldsymbol{H}\boldsymbol{M}_{j-1,j}$
    end if
    if $\delta h^2_{j-1,j-1} > h^2_{j-1,j} + h^2_{j,j}$ **then**
      $\widetilde{\boldsymbol{H}} \leftarrow \boldsymbol{G}^T_{j-1,j}\boldsymbol{H}\boldsymbol{P}_{j-1,j}$
      $j \leftarrow \max\{j-1, 2\}$
    else
      for $i \leftarrow j-2$ to 1 **do**
        if $|h_{i,j}| > \frac{1}{2}|h_{i,i}|$ **then**
          $\widetilde{\boldsymbol{H}} \leftarrow \boldsymbol{H}\boldsymbol{M}_{i,j}$
        end if
      end for
      $j \leftarrow j+1$
    end if
  end while
end function

With this procedure, the second condition, as described by (15b), is met, since

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

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

$$\tilde{h}_{1,1} \ll \tilde{h}_{2,2} \ll \ldots \ll \tilde{h}_{n,n}\,. \tag{21}$$

Note that the algorithm employed to find the ILS solution (see the next section) performs a depth-first search—the candidate ILS solutions are found by aggressively searching the n-dimensional space. Ordering the diagonal entries of $\widetilde{\boldsymbol{H}}$ reduces the number of nodes to be considered in the early steps of the search procedure, decreasing the total number of nodes to be explored.

The pseudocode of the LLL reduction procedure is provided in Algorithm 1, and an illustrative example of the effect of the LLL algorithm on $\boldsymbol{H}$ is presented in Fig. 4.

Therefore, with transformations (17) and (20) applied to $\boldsymbol{H}$, a matrix of the following form

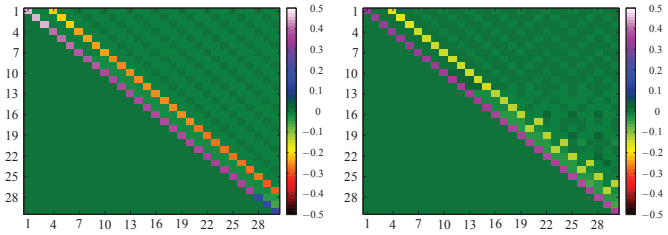
$$\widetilde{\boldsymbol{H}} = \boldsymbol{V}^T \boldsymbol{H} \boldsymbol{M} \tag{22}$$

is generated, with $\boldsymbol{V} \in \mathbb{R}^{n \times n}$ being an orthonormal matrix and $\boldsymbol{M} \in \mathbb{Z}^{n \times n}$ being unimodular (i.e. $\det \boldsymbol{M} = \pm 1$). With this, the ILS problem (13) can be rewritten as

$$\underset{\widetilde{\boldsymbol{U}}(k)}{\text{minimize}} \ \ || \widetilde{\boldsymbol{U}}_{\text{unc}}(k) - \widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{U}}(k)||_2^2 \tag{23a}$$

$$\text{subject to} \ \ \boldsymbol{U}(k) \in \mathbb{U} \tag{23b}$$

$$||\Delta \boldsymbol{u}(\ell)||_\infty \leq 1\,,\ \forall\, \ell = k, \ldots, k+N-1\,, \tag{23c}$$



(a) The $\boldsymbol{H}$ matrix. (b) The $\widetilde{\boldsymbol{H}}$ matrix.

Fig. 4: Visualization of the lattice generator matrix $\boldsymbol{H}$ for $n = 30$ (a) before, and (b) after the LLL reduction algorithm.

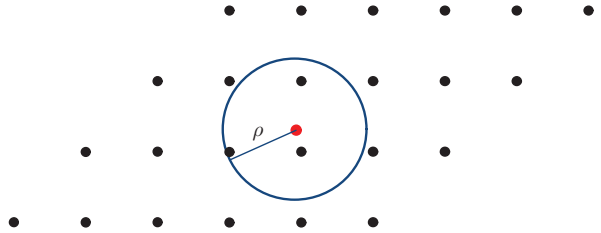


Fig. 5: The principle of the sphere decoder: A (two-dimensional) sphere of radius ρ (shown in blue line) centered at the unconstrained solution (shown as red solid circle) includes several integer points of the lattice (shown as black solid circles). One of these points is the integer solution of the ILS problem.

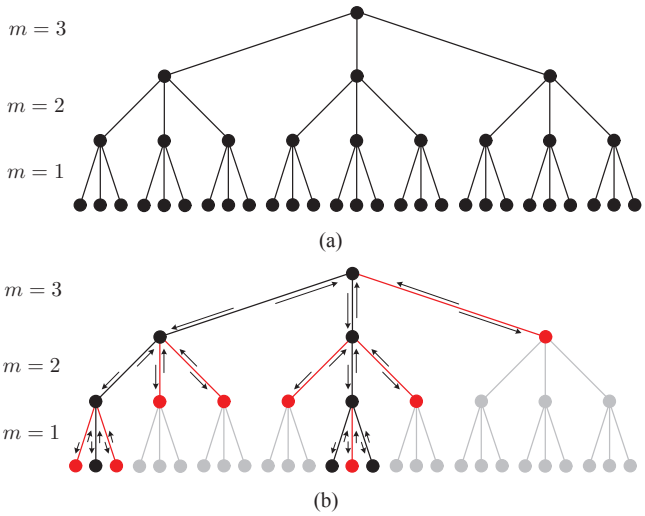


Fig. 6: (a) A typical search tree for the ILS problem (23), assuming a three-dimensional problem ($n = 3$). The index $m = \{1, \ldots, n\}$ refers to the layer in the search tree. As the search progresses, the dimension of the sphere is reduced from $m = n$ to a one-dimensional sphere. (b) Search tree with nodes explored by the sphere decoder (shown as black solid circles). The nodes that do not lie within the m-dimensional sphere are shown as red solid circles, whereas nodes that are not evaluated at all are depicted as gray solid circles. The direction of the search process is shown with black arrows; to find the optimal solution nodes are visited with a direction from left to right, and from the higher dimensional layers to the lower ones, until reaching a dead end or the bottom level, where backtracking occurs.

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

B. Search for the Optimal Solution

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 [5]. This algorithm is a variant

Algorithm 2 Sphere Decoder

function $\widetilde{\boldsymbol{U}}^* = \text{SPHDEC}(\boldsymbol{U},\, d^2,\, i,\, \rho^2,\, \widetilde{\boldsymbol{U}}_{\text{unc}})$
  for each $u \in \mathcal{U}$ **do**
    $U_i \leftarrow u$
    $d'^2 \leftarrow ||\, \widetilde{\boldsymbol{U}}_i - \widetilde{\boldsymbol{H}}_{(i,i:n)}\, \boldsymbol{U}_{i:n}||_2^2 + d^2$
    if $d'^2 \leq \rho^2$ **then**
      if $i > 1$ **then**
        $\text{SPHDEC}(\boldsymbol{U}, d'^2, i-1, \rho^2, \widetilde{\boldsymbol{U}}_{\text{unc}})$
      else
        if $\boldsymbol{U}$ meets (23c) **then**
          $\widetilde{\boldsymbol{U}}^* \leftarrow \boldsymbol{U}$
          $\rho^2 \leftarrow d'^2$
        end if
      end if
    end if
  end for
end function

of the Fincke and Pohst sphere decoding algorithm [12], which exploits the structure of the switching vector $\boldsymbol{U}$. According to the sphere decoding principle, the optimal solution lies within a hypersphere (n-dimensional sphere) of radius ρ, see Fig. 5. As mentioned in Section III-A, these algorithms are depth-first search algorithms that find the optimal solution by traversing a search tree (Fig. 6(a)) in a sequential manner until reaching a dead end or the bottom level, i.e. the one-dimensional nodes; in such a case they backtrack to examine unexplored nodes in higher layers, see Fig. 6(b).

To exploit the structure of $\widetilde{\boldsymbol{H}}$ and to avoid unnecessary computations, the sphere decoding algorithm is implemented as a "from bottom to top" search algorithm. This is in contrast to [5], where the search procedure is performed in a top-to-bottom manner. The pseudocode of the proposed algorithm is summarized in Algorithm 2.

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 [13], [14], which is a suboptimal solution to the ILS problem[5]. For the ILS problem (13), the Babai estimate can be easily computed by simply rounding the unconstrained solution $\boldsymbol{U}_{\text{unc}}$ to the closest integer vector

$$\boldsymbol{U}_{\text{bab}}(k) = \lfloor \boldsymbol{U}_{\text{unc}}(k) \rceil\,. \tag{24}$$

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

$$\rho(k) = ||\, \widetilde{\boldsymbol{U}}_{\text{unc}}(k) - \widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{U}}_{\text{bab}}(k)||_2\,, \tag{25}$$

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

It can be shown that the entry u_j of $\boldsymbol{U}$ is bounded between [7]

$$\left\lceil \frac{-\rho + \tilde{u}_{\text{unc}_j}}{\tilde{h}_{j,j}} \right\rceil \leq u_j \leq \left\lfloor \frac{\rho + \tilde{u}_{\text{unc}_j}}{\tilde{h}_{j,j}} \right\rfloor\,, \tag{26}$$

[5]In [5] an alternative calculation method for $\rho(k)$ is proposed, which exploits the receding horizon policy of MPC. More specifically, the previously computed optimal switching sequence is shifted by one step back in time.

since

$$\rho^2(k) \geq || \widetilde{\boldsymbol{U}}_{\text{unc}}(k) - \widetilde{\boldsymbol{H}}\widetilde{\boldsymbol{U}}(k)||_2^2 \,. \tag{27}$$

If condition (26) holds, then the solution set is nonempty and at least one lattice point exists.

Finally, it should be pointed out that when the LLL reduction algorithm is used for the transformation of the lattice, the probability of the Babai estimate $\boldsymbol{U}_{\text{bab}}$ being equal to the optimal solution $\boldsymbol{U}^*$ is increased, as shown in [15]. 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 enumerated are less.

IV. Analysis of the Computational Complexity

In this section, the computational complexity of the proposed MPC algorithm is analyzed. When implementing this algorithm on a microprocessor or a field-programmable gate array (FPGA), one needs to ensure that the algorithm always converges within the available time interval. Therefore, this analysis focuses on the worst-case number of operations.

The analysis of the computational burden of the direct MPC algorithm is divided into two parts: (a) the formulation of the ILS problem (13) and the preprocessing stage, based on the LLL reduction procedure (Algorithm 1), and (b) the optimization stage, i.e. the sphere decoding algorithm (Algorithm 2). The reason for splitting the analysis into two parts is that the formulation of the ILS problem and the preprocessing stage—under the given assumptions, i.e. the lattice generator matrix $\boldsymbol{H}$ is time-invariant—can be done offline, whereas the optimization stage is executed in real time. The second part of the algorithm thus determines the online computational demand, while the algorithm's first part is given for reasons of completeness.

A. Complexity of the ILS Problem Formulation and Preprocessing Stage

For the formulation of the ILS problem (13), the computational effort is concentrated in the factorization of $\boldsymbol{Q}$ to compute the lattice generator matrix $\boldsymbol{H}$. For the Cholesky factorization of $\boldsymbol{Q}$ $(1/3)n^3$ floating-point operations (flops)[6] are required [10].

For the LLL reduction algorithm, as shown in [16], the average complexity is $O(n^3 \log n)$. This upper bound on the number of flops required is derived by 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, though, 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 (15a) and (15b), see Fig. 4.

[6]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: Average and maximum number μ of nodes that are evaluated by the sphere decoder (Algorithm 2) without and with the LLL reduction algorithm (Algorithm 1), depending on the length of the prediction horizon N.

Prediction Horizon N	Without LLL		With LLL	
	avg(μ)	max(μ)	avg(μ)	max(μ)
1	3.18	7	3.18	7
2	7.88	19	6.40	14
3	14.01	39	9.58	19
4	22.48	87	12.86	27
5	33.02	148	12.21	44
7	63.18	690	23.05	61
10	132.97	831	36.21	141

B. Complexity of the Sphere Decoding Algorithm

The ILS problem, either in its initial form (13) or in its final form (23), is known to be NP-hard [14], [17]. 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 average and the maximum number of examined nodes μ are 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 assuming a drive system with a three-level inverter as in Fig. 1, the parameters of which can be found in Section V. The sampling interval is chosen as $T_s = 25\,\mu$s, and the weighting factor λ_u is tuned such that a switching frequency of about 300 Hz results.

Table I summarizes the computational burden of the sphere decoder (and consequently of the proposed MPC algorithm, since only these operations are performed in real time) 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. For the horizon $N = 10$, for example, the average number of nodes is reduced by 73%, while the maximum number of nodes is reduced by 83%. The latter number is of importance for a real-time implementation, indicating that the worst-case computations are reduced by 83%.

Even though the number of nodes explored by the sphere decoder provides an indication of the expected complexity, a more precise analysis is to be performed to define an upper bound on the expected operations to be performed in real time. First, 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. For this, $n(n+1)/2$ multiplications and $n(n-1)/2$ additions are required, i.e. n^2 flops in total.

Second, the nodes explored by the search algorithm are of different dimension, i.e. they belong to different layers of the search tree (see Fig. 6). Hence, the operations performed by the sphere decoder are a function of the dimension of the node examined. Specifically, for an m-dimensional node,

TABLE II: Average and maximum number of operations N_t performed in real time during the optimization stage, i.e. flops required to compute the unconstrained solution, as well operations performed by the sphere decoder (Algorithm 2). Two cases are presented—without and with the LLL reduction algorithm (Algorithm 1)—depending on the length of the prediction horizon N.

Prediction	Without LLL		With LLL	
Horizon N	avg(N_t)	max(N_t)	avg(N_t)	max(N_t)
1	44.73	99	44.73	99
2	165.39	369	138.72	291
3	371.04	915	278.23	501
4	698.95	2905	466.43	897
5	1156.61	5667	701.98	1587
7	2606.68	10275	1320.87	3030
10	6580.55	42147	2714.86	8268

$n-m+1$ additions[7], one subtraction, and $n-m+2$ multiplications are performed, whereas no division is required. Nonetheless, since $\boldsymbol{U} \in \mathbb{U}$ the result of the $n-m$ multiplications performed at the mth dimension 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 each node explored—and assuming that the constraint (23c) is inactive—the two remaining nodes are evaluated to check if they lie inside the hypersphere.

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

$$\begin{aligned} N_a &= \frac{n(n-1)}{2} + 3\left(\mu - 1 + \sum_{\substack{\text{nodes}\\ \text{explored}}} (n-m)\right), \\ N_s &= 3\mu\,, \\ N_m &= \frac{n(n+1)}{2} + 3\mu\,. \end{aligned} \tag{28}$$

To derive an upper bound on the operations performed, the 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), (b) $\boldsymbol{U} \in \mathbb{U} \setminus \{0\}^{N\times 3}$, since multiplications with zero result in a decrease in the number of additions, (c) constraint (23c) is inactive, and (d) only one out of the three m-dimensional nodes with the same $(m+1)$-dimensional root node is explored. Thus, the maximum of the total number of operations $N_t = N_a + N_s + N_m$ serves as an upper bound on the real-time computations.

The average number of operations performed in real time avg(N_t), as well as the respective upper bound $\max(N_t)$, are outlined in Table II. Again, the same two cases as in Table I are examined, i.e. the case where the solution $\boldsymbol{U}^*$ lies in a hypersphere in the lattice $\mathcal{L}(\boldsymbol{H})$, and the case where the solution $\tilde{\boldsymbol{U}}^*$ lies in a hypersphere in the lattice $\mathcal{L}(\tilde{\boldsymbol{H}})$. In addition, the dimensions of the lattices examined are the same as those presented in Table I.

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

V. Performance Evaluation

The simulation results presented in this section relate to a MV drive (see Fig. 1) consisting of a three-level NPC inverter with the constant dc-link voltage $V_{\text{dc}} = 5.2\,\text{kV}$ and a fixed neutral point N. A squirrel cage IM 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 and a 0.25 p.u. total leakage inductance. For all cases examined, the control algorithm was operated with the sampling interval $T_s = 25\,\mu\text{s}$. The results are shown in the p.u. system.

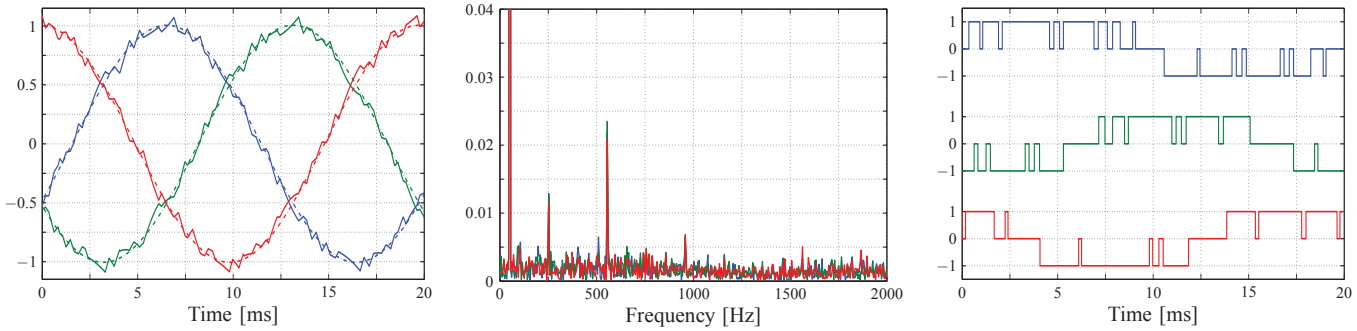
The steady-state performance of the drive is shown in Fig. 7. The horizon $N = 10$ case is investigated; the weighting factor $\lambda_u = 0.1$ is chosen, such that a switching frequency of approximately $300\,\text{Hz}$ is obtained. The three-phase stator current waveforms and their references are illustrated over one fundamental period in Fig. 7(a). Moreover, the resulting current spectrum is shown in Fig. 7(b); the total harmonic distortion (THD) of the current is 4.95%. Finally, Fig. 7(c) depicts the three-phase switching sequence.

To highlight the computational efficiency of the proposed direct MPC algorithm, it is compared with the exhaustive enumeration algorithm, which is typically used in the field of power electronics to solve integer programming problems formulated in the framework of MPC, as well as with the state-of-the-art sphere decoding algorithm introduced in [5]. Table III shows the average and the maximum number of examined nodes μ for different prediction horizons.The resulting current THD is shown to highlight the fact that with longer prediction intervals the closed-loop performance of the system can be improved.

As can be seen, the sphere decoding algorithm with the modifications proposed in this work significantly reduces the number of nodes examined, even compared with the approach proposed in [5]. Particularly, the maximum number of nodes—which is of more interest since it indicates the worst-case scenario—is reduced by about 45%.

VI. Conclusions

This paper proposes a direct model predictive control (MPC) scheme for current reference tracking with a low computational burden. The algorithm introduced in [5] is refined to further reduce the computational complexity of the underlying integer least-squares (ILS) problem. Modifications in the preprocessing stage, as well as in the sphere decoder are proposed. Specifically, in the preprocessing phase, a lattice reduction algorithm is implemented that reformulates the underlying ILS problem such that it can be solved more efficiently, i.e. it transforms the optimization problem to a well-conditioned one. In the second phase, i.e. the optimization stage, the initial radius of the hypersphere is calculated in an effective way based on the Babai estimate. This ensures that the solution set is always nonempty and that the radius is as small as possible. Thanks to all the aforementioned modifications, the computational burden can be reduced by up to 45% for long horizons and a three-level converter, compared to that required for the search algorithm proposed in [5].



(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 $\boldsymbol{u}$.

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 is approximately 300 Hz and the current THD 4.95%.

TABLE III: Average and maximum number μ of nodes evaluated by (a) the exhaustive search algorithm, (b) the sphere decoder in [5], and (c) the proposed algorithm. The resulting current THD for each prediction horizon length is also shown.

Prediction	Exhaustive Search		Sphere Decoder [5]		Proposed Approach		
Horizon N	avg(μ)	max(μ)	avg(μ)	max(μ)	avg(μ)	max(μ)	THD %
1	17.25	37	3.18	7	3.18	7	5.76
2	260	517	6.49	16	6.40	14	5.65
3	3,580	7,371	9.74	24	9.58	19	5.43
4	53,389	103,518	13.02	31	12.86	27	5.37
5	717,000	1,455,000	16.47	50	12.21	44	5.29
7	$> 1 \cdot 10^8$	$> 3 \cdot 10^8$	23.64	99	23.05	61	5.09
10	$> 1 \cdot 10^{12}$	$> 2 \cdot 10^{12}$	38.93	255	36.21	141	4.95

VII. Appendix

The matrices $\boldsymbol{\Upsilon}$, $\boldsymbol{\Gamma}$, $\boldsymbol{S}$, and $\boldsymbol{\Xi}$ in (7) are

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

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

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

References

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

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

[3] 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.

[4] 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.

[5] T. Geyer and D. E. Quevedo, "Multistep direct model predictive control for power electronics—Part 1: Algorithm," in *Proc. IEEE Energy Convers. Congr. Expo.*, Denver, CO, Sep. 2013, pp. 1154–1161.

[6] ——, "Multistep direct model predictive control for power electronics—Part 2: Analysis," in *Proc. IEEE Energy Convers. Congr. Expo.*, Denver, CO, Sep. 2013, pp. 1162–1169.

[7] 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.

[8] 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.

[9] 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.

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

[11] G. Strang, *Linear Algebra and its Applications*, 4th ed. Stamford, CT: Cengage Learn., 2005.

[12] 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.

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

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

[15] 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.

[16] 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.

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